Adaptive Labeling for Deep Learning to Hash

Huei-Fang Yang[1] Cheng-Hao Tu[2] Chu-Song Chen[2,3,4]
[1]Dept. CSIE, National University of Kaohsiung, Taiwan
[2]Institute of Information Science, Academia Sinica, Taiwan
[3]Research Center for Information Technology Innovation, Academia Sinica, Taiwan
[4]MOST Joint Research Center for AI Technology and All Vista Healthcare
hfyang@nuk.edu.tw r04922023@ntu.edu.tw song@iis.sinica.edu.tw

Abstract

Hash function learning has been widely used for large-scale image retrieval because of the efficiency of computation and storage. We introduce AdaLabelHash, a binary hash function learning approach via deep neural networks in this paper. In AdaLabelHash, class label representations are variables that are adapted during the backward network training procedure. We express the labels as hypercube vertices in a K-dimensional space, and the class label representations together with the network weights are updated in the learning process. As the label representations (or referred to as codewords in this work), are learned from data, semantically similar classes will be assigned with the codewords that are close to each other in terms of Hamming distance in the label space. The codewords then serve as the desired output of the hash function learning, and yield compact and discriminating binary hash representations. AdaLabelHash is easy to implement, which can jointly learn label representations and infer compact binary codes from data. It is applicable to both supervised and semi-supervised hash. Experimental results on standard benchmarks demonstrate the satisfactory performance of AdaLabelHash.

1. Introduction

Hashing techniques construct a mapping from high-dimensional data to low-dimensional binary codes from training samples, with an aim to preserve semantic similarity of samples in the Hamming space. The learned binary codes are fast in feature comparison, storage efficient, and well suitable for large-scale vision problems such as image or video retrieval. Over the past decade, great amounts of methods have been proposed to learn compact hash codes. These traditional approaches seek hash functions based on the handcrafted features [10, 12, 19, 22, 29]. In recent years, deep learning-based hashing approaches have become popular and attracted much attention [4, 8, 13, 14, 17, 31, 37]. Deep networks enable end-to-end learning, in which the image representations and hash codes are jointly evolved. Consequently, deep hashing approaches have achieved superior retrieval performance.

In deep supervised hashing in which the semantic labels provide supervision, one can train a classifier via neural networks to find hash functions during the classifier learning process. This could result in a single model that produces both the classification results and the binary hash codes. One representative approach is SSDH [31]. It seeks the hash mapping on a classification model by adding a latent layer right before the classification layer. Outputs of the latent layer are enforced to be close to binary, and the classification is assumed to rely on the latent concepts. The quantized binary features of the latent layer then serve as the hash codes. SSDH unifies classification and retrieval in a single learning model and enables end-to-end training of hash codes in a pointwise fashion. However, SSDH needs pre-defined class-label representations to guide the training.

This paper introduces AdaLabelHash for hash mapping learning, which leverages deep networks to seek appropriate label representations for classification. In AdaLabelHash, the semantic labels are trainable variables. Instead of encoding the labels beforehand as done in standard supervised learning, we encode the labels as vertices of a K-dimensional unit-hypercube that are variables during training. As the label representations are free variables, no biases or constraints are imposed. After learning, similar classes can be expressed with the labels found in the unit-hypercube vertices.

Specifically, we assume that each label has its own representation codeword that is the driving force behind hash coding. The codewords are attractors that draw relevant samples close in the label space and also are repulsors that keep irrelevant samples apart. As shown in Figure 1, the label representations $\{v\}$ are expressed as a soft sign map-

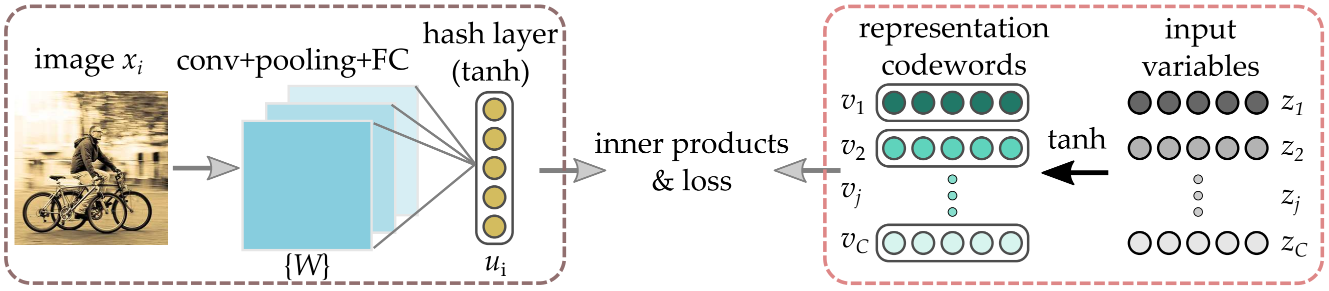

Figure 1. AdaLabelHash assumes that C unknown codewords $\{v_j\}$, each of which expresses a class label, are variables to be recovered, and hash coding is guided by these codewords. Each codeword attracts (repulses) data of similar (dissimilar) semantic meanings to the concept it represents. In the left part, conv+pooling+FC denotes a neural network consisting of convolution layers, max-pooling and fully connected layers, and their weights $\{W\}$ are unknowns to be learned too. The representation codewords $\{v_j\}$ are generated by applying a soft sign function ($\tanh$) to dummy variables $\{z_j\}$, the input to the right-hand-side network. As the value of each entry in $\{v_j\}$ is within $[-1, 1]$, it serves as an approximation of unit-hypercube vertex. Both the dummy free variable $\{z_j\}$ in the right part and the network weights $\{W\}$ in the left part are variables to be found during the learning process via the back-propagation algorithm.

ping ($\tanh$) of the variable inputs $\{z\}$ of a neural network. Given the input image, the network weights W and variable inputs z are jointly learned via the back-propagation training procedure, which generate the hash codes u and label representations v.

Employing unknown inputs as variables has been utilized for visualizations (*e.g.*, [23] and DeepDream), image style transfer [9] and optimizing generative models [1]. We exploit this idea for learning hash codes and afford a length-K binary representation per class. Besides, with the label representations, we can generalize AdaLabelHash to handle semi-supervised hashing via self-training [25].

The main characteristics of this paper are as follows:

- A new approach is introduced that explores the label representations (or codewords) of semantic categories as the unknown inputs for classification via deep models. These label representations guide the hash coding.
- Our approach can learn similarity-preserving hash codes and discover the label representations simultaneously during the learning process, with no needs of prior assumptions for the free input variables.
- Our approach is flexibly applicable to both single-label and multi-label data, and extendable for semi-supervised hashing.

2. Related Work

Learning-based hashing approaches can be classified into three categories: unsupervised, supervised, and semi-supervised hashing. Unsupervised hashing constructs hash functions that preserve the similarity between data points in the original input space based solely on samples without labels, including SH [29], ITQ [10], and deep learning-based approaches such as DH [18] and DeepBit [15, 16]. Among the aforementioned approaches, in addition to similarity-preserving, DeepBit further learns binary descriptors that are rotationally invariant.

Supervised hashing learns binary codes by fully exploiting annotated information provided in the data. The supervision can be pairwise similarity relations (*e.g.*, BRE [12] and MLH [19]) or class labels (*e.g.*, ITQ-CCA [10] and RSH [32]). Traditional methods learn hash functions to encode hand-crafted visual descriptors into binary codes. Recent deep hashing approaches have shown superior performance. This is because deep networks can learn image representations and hash functions simultaneously, and the learned binary codes exhibit better discriminability. One of the early deep hashing approaches is CNNH [30]. It is a two-stage approach, where the first stage learns approximate hash codes and the second uses a deep network to learn a mapping from the training data to hash codes. However, it has a limitation that hash code learning and data representation learning cannot benefit each other. DNNH [13] overcomes this limitation by jointly optimizing hash coding and feature representation learning within one network. Later, much of the work has been proposed. DSRH [36] proposes to utilize the multilevel similarity information to guide the learning of hash functions. DHN [37] and DPSH [14] leverage pairwise label similarity to optimize cross-entropy loss. To learn concentrated hash codes, DCH [3] designs a modified pairwise cross-entropy loss based on Cauchy distribution. DVSQ [4] learns a visual-semantic embedding by a two-stream network, one taking as input the labels and the other the data, where the labels are generated by a skip-gram model, *e.g.*, word2vec. The recent HashGAN [2] utilizes a generative model to synthesize images conditioned on the pairwise similarity information for performance improvement. On the other hand, assuming that classification depends on a set of *on* and *off* attributes, [17] learns binary codes that preserve label semantics. SSDH [31] treats binary codes as hidden concepts that govern classification and learns discriminative codes complying with semantic labels as well as hash code properties. Besides, some recent studies [6, 8, 24] have been focused on handling the discrete constraint imposed on the networks. HashNet [6] solves this problem by gradually changing a smoothed objective function to more non-smooth one during the course of training. DMDH [8] minimizes the discrepancy between the continuous hash codes and the desired discrete binary codes. The above approaches still adopt a continuous relaxation to the discrete constraint. One can directly train on the binary codes via policy gradient, as done in PGDH [33].

Due to the high cost to obtain extensive annotations and the availability of abundant data without complete annotations, semi-supervised hashing algorithms, *e.g.*, SSH [27], SSTH [28] and MLAGH [11] have been proposed, which leverage the unlabeled samples in addition to the labeled ones to perform hash code learning. The learned binary codes preserve the semantic similarity of the labeled data and the underlying data structures. Deep semi-supervised hashing has received attention more recently. Semi-SDH [34] utilizes a graph embedding to discover the similarity between labeled and unlabeled data. DSH-GANs [20] employs generative adversarial models to generate synthetic images for enriching the diversity of training samples. The hash functions are learned from both the real and the synthetic data.

Previous deep hashing approaches do not provide representation codewords learned for semantic categories. Our AdaLabelHash not only affords explicit representations for semantic labels but can also capture the correlation between labels as shown in Discussion.

3. Adaptive Labeling for Deep Hashing

Let $\mathrm{X} = \{x_i\}_{i=1}^N$ denote a set of N training images labeled with C classes. Each image belongs to one class (single label case) or several classes (multi-labels case). Denote $S = \{s_{ij}\}$ to be the association matrix, with $s_{ij} = 1$ if x_i belongs to class j and $s_{ij} = 0$ otherwise, $j = 1 \cdots C$. Our goal is to learn a mapping $\Omega : \mathrm{X} \rightarrow \{-1, 1\}^{K \times N}$ that projects a sample x_i into a K-bit hash code $u_i = \Omega(x_i)$.

We use a deep network to learn the mapping, $\mathcal{F}(x_i, \mathcal{W}) \in \mathbb{R}^K$, where $\mathcal{W}$ denotes the weights of the network $\mathcal{F}$ and the output is a K-dimensional vector. The hash codes are then obtained by taking the sign of the network outputs, $u_i = \operatorname{sgn}(\mathcal{F}(x_i, \mathcal{W}))$, and $\operatorname{sgn}(\cdot) \in \{\pm 1\}^K$ is the sign function.

In common supervised learning, a C-dimensional vector is used to represent the C classes. For example, to learn a classifier in the single-label case, the j-th class is often represented as a length-C one-hot vector, with the j-th element being 1 and the others being 0. In our work, to learn a hash function mapping, the classes are not expressed as a constant C-dimensional vector, but variable vectors in K-dimensional space, where K is the length of the hash-code.

More specifically, we assume that each class has its own codeword in the K-dimensional space. The set of codewords is denoted by $\mathrm{V} = \{v_j\}_{j=1}^C \in \{-1, 1\}^{K \times C}$, where v_j is a K-bit binary codeword, or a vertex in the K-dimensional hypercube. When the codewords are known beforehand, one can directly employ them as the learning goal and optimize a loss derived based on the fixed, given codewords. However, without any prior knowledge about the underlying data distribution, it is unlikely to obtain representation codewords that well partition the feature space. To avoid the need to hand-craft the representation codewords, we instead treat the codewords as part of the network parameters to be learned. We assume that the codewords V are directly derived from a set of dummy variables $Z = \{z_j\}_{j=1}^C \in \mathbb{R}^{K \times C}$, where $v_j = \operatorname{sgn}(z_j)$. The codewords act as attractors and repulsors during learning. The learning objective is to make the binary code u_i of sample x_i and a codeword v_j close if $s_{ij} = 1$, and simultaneously keep u_i and v_j as apart as possible if $s_{ij} = 0$.

The closeness between a pair of binary codes h_i and v_j is determined by their Hamming distance: $\operatorname{dist}_H(u_i, v_j) = \frac{1}{2}(K - \langle u_i, v_j \rangle)$, where $\langle \cdot, \cdot \rangle$ is the inner product. Hence, the inner product reflects the closeness between binary codes. We define the objective function of AdaLabelHash for N training samples as:

$$\min_{\mathcal{W},Z} L(\mathcal{W}, Z) = \sum_{i=1}^{N} \sum_{s_{ij}=1} \max(0, m - u_i^T v_j + \max(u_i^T v_t[s_{it} \neq 1])), \qquad (1)$$

where m is the margin, which defines the inner product of a similar sample-codeword pair should be at least m larger than the least dissimilar sample-codeword pair, and Iverson bracket indicator function when $[s_{ij} \neq 1]$ evaluates to 1 $s_{ij} \neq 1$ and 0 otherwise, with $\{s_{ij}\}$ the associatoin matrix between data and label. In our experiments, m is set to 1.

A problem occurs in the above formulation is that we cannot back-propagate the gradients to the unknowns $\mathcal{W}$ and Z due to the $v_j = \operatorname{sgn}(z_j)$ and $u_i = \operatorname{sgn}(\mathcal{F}(x_i, \mathcal{W}))$ functions. In AdaLabelHash, we use a soft sign function $\tanh(\cdot)$ to approximate the $\operatorname{sign}(\cdot)$ instead, as shown in Fig. 1. That is, $v_j = \tanh(z_j)$ and $u_i = \tanh(\mathcal{F}(x_i, \mathcal{W}))$. The similarity is measured by the real-valued hash codes u_i and codewords v_j and is given as $u_i^T v_j$. AdaLabelHash learns the hash coding $\mathcal{F}(x_i, \mathcal{W})$ and the codewords V simultaneously. That is, both the network weights and class label representations are adjusted via back propagation. Because the label representations (codewords) are learned from data, they are expected to better capture the underlying data structure than the hand-crafted labeling, where semantically similar classes can be represented by closer codewords in the K-dimensional space, where K is the hash-code length set for hash-function learning.

Once trained, the binary hash code of an input sample x can be efficiently obtained by converting the sample through the network and quantizing the activations of the hash layer, $h = \mathrm{sgn}(u) = \mathrm{sgn}(\mathcal{F}(x_i, \mathcal{W}))$. One may introduce a quantization error in the learning objective as in [6, 31], so that the learned codewords are more binarized. Currently, we have not use the quantization loss term in the formulation yet, but AdaLabelHash can incorporate it easily if needed.

4. Semi-supervised Hashing via Self-training

Leveraging the representation codewords, AdaLabelHash can be extended to semi-supervised learning, where the input data are provided with or without label annotations. Our method follows the self-training principle [25]. We first train AdaLabelHash on the labeled samples to make the codewords possess some discrimination power. Then, we treat the probabilities obtained by a softmax over the inner products between the hash layer and the codewords as confidence values and assign pseudo labels to the unlabeled samples. We include only the unlabeled data with pseudo labels of high confidences (*i.e.*, confidence values greater than a threshold) in the labeled data. The newly formed training set (*i.e.*, labeled data and unlabeled data with assigned labels) is used to train the neural network. This process is repeated several times to complete the semi-supervised learning.

Each single-label image is associated with only one label, so one can assume that an unlabeled input sample is associated with the label of the highest probability. However, the number of labels associated with a multi-label sample is unknown. To deal with this issue, we take a conservative approach to avoid performance degradation caused by mis-assignment of the labels to the samples, where only the most confident labels are used as the pseudo labels for an input sample. We sort the labels according to their probabilities in a descending order to obtain a ranked list. The ranked list is divided into three disjoint sets, most-probable (top-k), least-probable (bottom-k), and uncertain (remaining). We assume that the input sample contains the most probable labels and do not contain the least probable labels, and treat the uncertain labels as missing. The losses and gradients of missing labels are set to zeros, so that they do not contribute to the neural network learning of semi-supervised hashing.

5. Experiments

In the section, we evaluate the proposed AdaLabelHash on the CIFAR-10 and NUS-WIDE under the supervised and semi-supervised settings. We also visualize the learned binary representations to see if the label representations capture the semantic similarity among categories.

Datasets. CIFAR-10[1], one of the most popular benchmarks for evaluating retrieval algorithms, consists of 60,000 32×32 color images in 10 classes. It is a single-label dataset, and each class contains 6,000 images. NUS-WIDE[2] contains about 270,000 flickr images, but we were able to collect only about 230,000 images following the given URLs. It is a multi-label dataset, in which each image is associated with one or multiple labels in 81 concepts. Following the settings in [13, 30], we use a subset of 162,289 images in the 21 most frequent concepts. We split the data for the supervised and semi-supervised settings as follows.

Supervised setting: This setting follows that in [31]. In CIFAR-10, we use 10,000 images (1,000 images per class) as the query set and use the remaining images to train the network and to form the retrieval database. In NUS-WIDE, 2,100 images (100 images per concept) are randomly selected to construct the query set, and the rest are used as training images as well as the retrieval database.

Semi-supervised setting: Following the settings in [34], in CIFAR-10, we randomly select 1,000 images (100 images per class) as the query set and use the remaining 59,000 images as the retrieval database. 5,000 images (500 images per class) are selected from the retrieval database as the labeled training set and the remaining 54,000 images are used as unlabeled training set. In NUS-WIDE, 2,100 images (100 images per concept) are randomly selected as the query set, and the rest are used as the retrieval database. The labeled training set contains 10,500 images selected from the retrieval database, and the unlabeled training set contains the rest images.

Table 1 shows the details of the two settings.

Evaluation Metrics. We evaluate the retrieval performance using mean average precision (mAP). It is defined as the mean of the average precision (AP) of all the queries, that is, $\mathrm{mAP} = \frac{1}{N}\sum_i^N \mathrm{AP}_i$, with N the number of query images. The AP_i of a query image i is given as

$$\mathrm{AP}_i = \frac{1}{M}\sum_{r=1}^{R} \mathrm{prec}(r) \odot \mathrm{rel}(r), \tag{2}$$

where M is the number of relevant images in the returned list, $\mathrm{prec}(r)$ denotes the precision at the top r returned images, and $\mathrm{rel}(r)$ indicates whether the rth retrieved image is relevant to the query or not. $\mathrm{rel}(r) = 1$ means the returned

[1]https://www.cs.toronto.edu/~kriz/cifar.html

[2]http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm

Table 1. The statistics of datasets used in the experiments.

	Supervised				Semi-supervised			
Dataset	# classes	training	database	query	labeled training	unlabeled training	database	query
CIFAR-10	10	50,000	50,000	10,000	5,000	54,000	59,000	1,000
NUS-WIDE	21	160,189	160,189	2,100	10,500	146,989	160,189	2,100

Table 2. The network architecture adopted in our experiments for hash coding. LRN denotes the local response normalization that implements the lateral inhibition. The ReLU is the activation function for all the layers, except for the hash layer that uses the $\tanh$.

Layer	Configuration
conv1	filter 64x11x11, stride 4, pad 0, LRN, maxpool 2
conv2	filter 256x5x5, stride 1, pad 2, LRN, maxpool 2
conv3	filter 256x3x3, stride 1, pad 1
conv4	filter 256x3x3, stride 1, pad 1
conv5	filter 256x3x3, stride 1, pad 1, maxpool 2
fc6	4096 nodes
fc7	4096 nodes
hash	K nodes with $\tanh$ activations

image is a relevance of the query and 0 otherwise. We use image labels to determine whether or not two images are relevant. When two images share at least one common label, they are considered relevant.

Implementation details. Our implementation uses Keras with a Tensorflow backend. We choose the CNN-F [7] as the network model, which consists of 5 convolutional layers, 2 fully-connected layers (fc6 and fc7) and one classification output layer. We remove the output layer and add in a hash layer of K nodes with $\tanh$ activations. Table 2 shows the detailed architecture. The weights of the first 7 layers are initialized with the weights pre-trained on ImageNet [21] and the weights of the hash layer are initialized by a Xavier uniform.

Besides image inputs, the other inputs to AdaLabelHash are input variables ($Z = \{z_j\}_{j=1}^{C}$) for learning label representations of semantic categories. Codeword learning is similar to word embeddings in natural language processing where words are mapped to real-valued vectors, which can be easily implemented by the Embedding layer provided in Keras. The entire AdaLabelHash network is optimized via back-propagation using stochastic gradient descent (SGD) with Nesterov momentum.

5.1. Results on CIFAR-10

Supervised experiments. We compare AdaLabelHash with 11 deep hashing approaches, including SSDH [31], DPSH [14], DMDH [8], DRSCH [35], DSCH [35], DSRH [36], DQN [5], CNNH [30], DNNH [13], DHN [37], and DVSQ [4]. As shown in Table 3, AdaLabelHash provides better retrieval results than the approaches compared. This suggests that joint learning of label representations and hash codes are potentially useful and the learned hash codes exhibit good discriminability.

Table 3. The mAPs of supervised methods on CIFAR-10.

Method	24 bits	32 bits	48 bits
AdaLabelHash	**0.930**	**0.946**	**0.934**
SSDH [31]	0.919	0.914	0.914
DPSH [14]	0.781	0.795	0.807
DVSQ [4]	—	0.730	0.733
DMDH [8]	—	0.719	0.732
DRSCH [35]	0.622	0.629	0.631
DSCH [35]	0.613	0.617	0.620
DSRH [36]	0.611	0.617	0.618
DQN [5]	0.558	0.564	0.580
DHN [37]	0.594	0.603	0.621
DNNH [13]	0.566	0.558	0.581
CNNH [30]	0.511	0.509	0.522

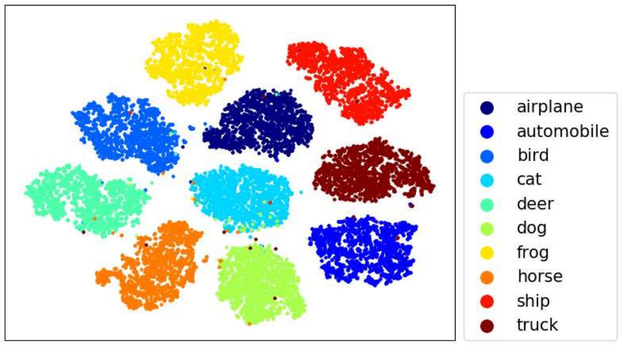

Figure 2. Visualization of the learned binary codes of AdaLabelHash by t-SNE on CIFAR-10 when the code length is 48.

Visualization of learned binary codes. Figure 2 shows the t-SNE [26] visualization of the binary codes learned by AdaLabelHash on CIFAR-10. As can be seen, AdaLabelHash can learn a representation space where images of the same semantic content form a group and the boundaries between different semantic groups are prominent. Besides, we observe that semantic groups (*e.g.*, bird, cat, dog) that share common higher-level abstraction (*e.g.*, animals) tend to reside closer in the feature space; therefore different higher-

Table 4. The mAPs of semi-supervised methods on CIFAR-10.

Method	24 bits	32 bits	48 bits
AdaLabelHash	0.806	**0.816**	**0.845**
Semi-SDH [34]	**0.813**	0.812	0.814
DSH-GANs [20]	0.781	0.787	0.802
MLAGH [11]	0.556	0.542	0.528

level abstractions reside in the different parts of the feature space. For instance, the left side of the figure is animals whereas the right side is with transportation concept. The representation structures are captured by AdaLabelHash as there are no additional high-level semantic priors, such as relationships among labels, imposed during learning.

Semi-supervised experiments. In Table 4, we compare AdaLabelHash to semi-supervised hashing approaches including Semi-SDH [34], DSH-GANs [20], and MLAGH [11]. The results show that AdaLabelHash outperforms almost all approaches at different code-length settings, except for Semi-SDH at a shorter code length (24 bits). The reason could be that Semi-SDH exploits an online graph approach, which can better capture the neighboring information between the unlabeled images and the label images. In contrast, AdaLabelHash uses a simpler approach but yields better results in overall.

5.2. Results on NUS-WIDE

Supervised experiments. Like the evaluation done on CIFAR-10, AdaLabelHash is also compared to several deep supervised hashing approaches on NUS-WIDE. As reported in Table 5, AdaLabelHash performs more favorably against almost all the approaches compared, except the SSDH on the fewer-bits case at code length 24. We think the reason is that a longer code would make the representation learning easier. Nevertheless, the results generally show that AdaLabelHash is effective on obtaining competitive results for hasing on mult-label cases. Note that the HashNet [6] and HashGAN [3] are not included in comparison because they adopt a different setting of using all the 81 concepts (labels). Although the results cannot be compared directly due to different settings, we still show their performance as a reference in the following. HashNet yields mAPs of 69.9% and 71.1% when the code lengths are 32 and 48, respectively. HashGAN attains 74.4% at code length 32 and 74.8% at the code length 48. Since more labels are used, their performance is lower than ours.

Semi-supervised experiments. Table 6 shows the comparison of AdaLabelHash with the semi-supervised hashing approaches, Semi-SDH [34] and MLAGH [11]. The MLAGH utilizes the CNN features, which are 300-d PCA-compressed features of the 4096-d features of fc6 (see Table 2) extracted from the pre-trained CNN-F. It can be observed that AdaLabelHash demonstrates superiority over other approaches across all bits. These results suggest again that AdaLabelHash can be readily applied to semi-supervised hashing and yield satisfactory performance.

Table 5. The mAPs of supervised methods on NUS-WIDE. The mAP is calculated based on top 50,000 returned images.

Method	24 bits	32 bits	48 bits
AdaLabelHash	0.758	**0.769**	**0.800**
SSDH [31]	**0.787**	0.750	0.782
DPSH [14]	0.722	0.736	0.741
DRSCH [35]	0.622	0.623	0.628
DSCH [35]	0.597	0.611	0.609
DSRH [36]	0.618	0.621	0.631

Table 6. The mAPs of semi-supervised methods on NUS-WIDE.

Method	24 bits	32 bits	48 bits
AdaLabelHash	**0.759**	**0.791**	**0.806**
Semi-SDH [34]	0.725	0.731	0.735
MLAGH [11]	0.701	0.712	0.717

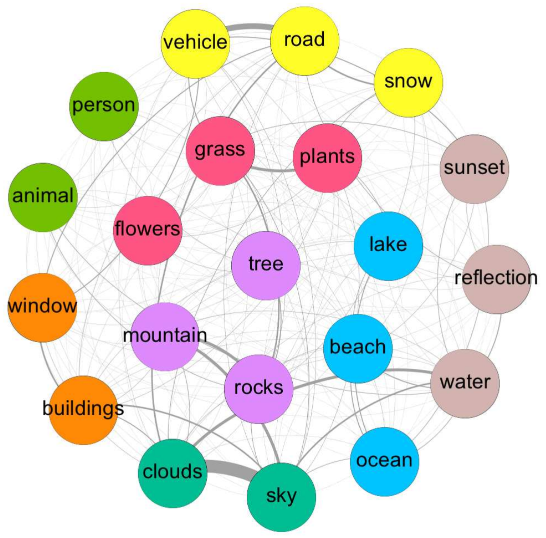

Figure 3. Relations among the learned representation codewords of NUS-WIDE. Thicker lines between codewords indicate smaller distances (*i.e.*, higher correlations) and different colors represent different concept groups.

Label relations discovered from data. Thus far, we have shown that AdaLabelHash can learn similarity-preserving hash codes and handle supervised and semi-supervised learning of hash functions. This is due largely to the label representations discovered from visual data. Also thanks to the explicit representations obtained, AdaLabelHash can find the correlation between labels. The labels may exhibit certain redundancy. We show that AdaLabelHash can remove redundancy among labels such that the labels of similar concepts are expected to have smaller distances between their codewords in the representation space.

Figure 3 shows the relations between the codewords learned from supervised AdaLabelHash for NUS-WIDE based on modularity optimization; thicker lines between codewords indicate smaller distances (*i.e.*, higher correlations) and different colors represent different concept groups found in the representation space. As can be seen, the label "clouds" has a higher correlation to a similar concept of "sky" than a different concept of "buildings". The labels "vehicle" and "road" are highly correlated while "sunset" and "person" are not. Additionally, the codewords of similar semantic labels form a cluster, *e.g.*, window and buildings belong to the same cluster; lake, beach, and ocean belong to another. Hence, the learned codewords exhibit the relations among semantic labels.

6. Conclusions

We have presented AdaLabelHash, a deep learning approach to hash function learning. AdaLabelHash can learn both label representations and neural-network weights simultaneously. The learned network can then be used to infer binary hash codes that exhibit good discriminating capability for image retrieval. Experimental results have demonstrated that AdaLabelHash achieves competitive retrieval performance for both supervised and semi-supervised hashing. The representation codewords learned from data via network training can reflect relations among the labels.

Acknowledgment

This work is supported in part by the Ministry of Science and Technology of Taiwan under grants MOST 108-2634-F-001-004 and MOST 107-2218-E-390-006-MY2.

References

[1] P. Bojanowski, A. Joulin, D. Lopez-Paz, and A. Szlam. Optimizing the latent space of generative networks. *CoRR*, abs/1707.05776, 2017. 2

[2] Y. Cao, B. Liu, M. Long, and J. Wang. HashGAN: Deep learning to hash with pair conditional wasserstein GAN. In *Proc. CVPR*, pages 1287–1296, 2018. 2

[3] Y. Cao, M. Long, B. Liu, and J. Wang. Deep cauchy hashing for hamming space retrieval. In *Proc. CVPR*, 2018. 2, 6

[4] Y. Cao, M. Long, J. Wang, and S. Liu. Deep visual-semantic quantization for efficient image retrieval. In *CVPR*, 2017. 1, 2, 5

[5] Y. Cao, M. Long, J. Wang, H. Zhu, and Q. Wen. Deep quantization network for efficient image retrieval. In *AAAI*, pages 3457–3463, 2016. 5

[6] Z. Cao, M. Long, J. Wang, and P. S. Yu. HashNet: Deep learning to hash by continuation. In *ICCV*, 2017. 3, 4, 6

[7] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*, 2014. 5

[8] Z. Chen, X. Yuan, J. Lu, Q. Tian, and J. Zhou. Deep hashing via discrepancy minimization. In *Proc. CVPR*, pages 6838–6847, 2018. 1, 3, 5

[9] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *CVPR*, pages 2414–2423, 2016. 2

[10] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(12):2916–2929, 2013. 1, 2

[11] H. Hu, K. Wang, C. Lv, J. Wu, and Z. Yang. Semi-supervised metric learning-based anchor graph hashing for large-scale image retrieval. *IEEE Trans. Image Processing*, 28(2):739–754, 2019. 3, 6

[12] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In *NIPS*, pages 1042–1050, 2009. 1, 2

[13] H. Lai, Y. Pan, Y. Liu, and S. Yan. Simultaneous feature learning and hash coding with deep neural networks. In *CVPR*, pages 3270–3278, 2015. 1, 2, 4, 5

[14] W. Li, S. Wang, and W. Kang. Feature learning based deep supervised hashing with pairwise labels. In *IJCAI*, pages 1711–1717, 2016. 1, 2, 5, 6

[15] K. Lin, J. Lu, C.-S. Chen, and J. Zhou. Learning compact binary descriptors with unsupervised deep neural networks. In *Proc. CVPR*, pages 1183–1192, 2016. 2

[16] K. Lin, J. Lu, C.-S. Chen, J. Zhou, and M.-T. Sun. Unsupervised deep learning of compact binary descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2018. 2

[17] K. Lin, H.-F. Yang, J.-H. Hsiao, and C.-S. Chen. Deep learning of binary hash codes for fast image retrieval. In *CVPRW on DeepVision: Deep Learning in Computer Vision*, pages 27–35, 2015. 1, 2

[18] J. Lu, V. E. Liong, and J. Zhou. Deep hashing for scalable image search. *IEEE Trans. Image Processing*, 26(5):2352–2367, 2017. 2

[19] M. Norouzi and D. J. Fleet. Minimal loss hashing for compact binary codes. In *ICML*, pages 353–360, 2011. 1, 2

[20] Z. Qiu, Y. Pan, T. Yao, and T. Mei. Deep semantic hashing with generative adversarial networks. In *ACM SIGIR*, pages 225–234, 2017. 3, 6

[21] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *Int'l J. Computer Visionl*, pages 211–252, 2015. 5

[22] F. Shen, C. Shen, W. Liu, and H. T. Shen. Supervised discrete hashing. In *CVPR*, pages 37–45, 2015. 1

[23] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2013. 2

[24] S. Su, C. Zhang, K. Han, and Y. Tian. Greedy Hash: Towards fast optimization for accurate hash coding in CNN. In *NeurIPS*, pages 806–815, 2018. 3

[25] I. Triguero, S. García, and F. Herrera. Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowledge and Information Systems*, 42(2):245–284, 2015. 2, 4

[26] L. van der Maaten and G. Hinton. Visualizing high-dimensional data using t-SNE. *Journal of Machine Learning Research*, 9: 25792605, 2008. 5

[27] J. Wang, S. Kumar, and S. Chang. Semi-supervised hashing for large-scale search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(12):2393–2406, 2012. 3

[28] Q. Wang, L. Si, and D. Zhang. Learning to hash with partial tags: Exploring correlation between tags and hashing bits for large scale image retrieval. In *ECCV*, pages 378–392, 2014. 3

[29] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, pages 1753–1760, 2008. 1, 2

[30] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan. Supervised hashing for image retreieval via image representation learning. In *AAAI*, pages 2156–2162, 2014. 2, 4, 5

[31] H.-F. Yang, K. Lin, and C.-S. Chen. Supervised learning of semantics-preserving hash via deep convolutional neural networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(2):437–451, 2018. 1, 2, 4, 5, 6

[32] X. Yuan, Z. Chen, J. Lu, J. Feng, and J. Zhou. Reconstruction-based supervised hashing. *Pattern Recognition*, 79:147–161, 2018. 2

[33] X. Yuan, L. Ren, J. Lu, and J. Zhou. Relaxation-free deep hashing via policy gradient. In *Proc. ECCV*, 2018. 3

[34] J. Zhang, Y. Peng, and J. Zhang. SSDH: Semi-supervised deep hashing for large scale image retrieval. *IEEE Trans. Circuits and Systems for Video Technology*, 29(1):212–225, 2019. 3, 4, 6

[35] R. Zhang, L. Lin, R. Zhang, W. Zuo, and L. Zhang. Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification. *IEEE Trans. Image Processing*, 24(12):4766–4779, 2015. 5, 6

[36] F. Zhao, Y. Huang, L. Wang, and T. Tan. Deep semantic ranking based hashing for multi-label image retreieval. In *CVPR*, pages 1556–1564, 2015. 2, 5, 6

[37] H. Zhu, M. Long, J. Wang, and Y. Cao. Deep hashing network for efficient similarity retrieval. In *AAAI*, pages 2415–2421, 2016. 1, 2, 5