Supervised Learning of Semantics-Preserving Hashing via Deep Neural Networks for Large-Scale Image Search

Huei-Fang Yang, Kevin Lin, Chu-Song Chen

***Abstract*—This paper presents a supervised deep hashing approach that constructs binary hash codes from labeled data for large-scale image search. We assume that semantic labels are governed by a set of latent attributes in which each attribute can be *on* or *off*, and classification relies on these attributes. Based on this assumption, our approach, dubbed supervised semantics-preserving deep hashing (SSDH), constructs hash functions as a latent layer in a deep network in which binary codes are learned by the optimization of an objective function defined over classification error and other desirable properties of hash codes. With this design, SSDH has a nice property that classification and retrieval are unified in a single learning model, and the learned binary codes not only preserve the semantic similarity between images but also are efficient for image search. Moreover, SSDH performs joint learning of image representations, hash codes, and classification in a pointwised manner and thus is naturally scalable to large-scale datasets. SSDH is simple and can be easily realized by a slight modification of an existing deep architecture for classification; yet it is effective and outperforms other unsupervised and supervised hashing approaches on several benchmarks and one large dataset comprising more than 1 million images.**

***Index Terms*—Image retrieval, supervised hashing, binary codes, deep learning, convolutional neural networks.**

I. Introduction

HASHING methods that construct similarity-preserving binary codes for image search have received great attention in the vision community [1]–[4]. The key principle in devising the hash functions is to map images of similar content to similar binary codes, which amounts to mapping the high-dimensional visual data into a low-dimensional Hamming (binary) space. Having done so, one can perform an approximate nearest-neighbor (ANN) search by simply calculating the Hamming distance between the binary vectors, an operation that can be done extremely fast.

Recently, learning-based hashing approaches have become popular as they leverage training samples in code construction and the learned binary codes are more efficient than the ones by locality sensitive hashing (LSH) [5]–[7] that maps similar images to the same bucket with high probability through random projections, makes no use of training data, and thus requires longer codes to attain high search accuracy. Among various learning-based hashing approaches, supervised hashing in which the supervised information (e.g., similarity between image pairs, or labels) is exploited during hash function learning can learn binary codes that better capture the semantic structure between image data. Although the supervised hashing approaches yield promising performance, many employ pairs or triplets of the training samples in the training phase and need a long computational time and a high storage cost for training, making them impractical for huge collections of images.

In this paper, by taking advantage of deep learning, we propose the supervised semantics-preserving deep hashing (SSDH) for learning binary codes from labeled training images. The idea behind SSDH is that image labels can be implicitly represented by a set of latent attributes (i.e., binary codes) and the classification is dependent on these attributes. On the basis of this idea, we construct the hash functions as a latent layer between image representations and classification outputs in a convolutional neural network (CNN), and the binary codes can be learned by the minimization of an objective function defined over classification error. This design yields a simple and effective deep network that unifies classification and retrieval in a single learning process and also encourages semantically similar images to have similar binary codes. Moreover, to make the hash codes more compact, we impose additional constraints on the learning objective to make each hash bit carry as much information as possible. The overview of SSDH is illustrated in Figure 1. Experimental results on several benchmarks show that our SSDH provides superior performance over other hashing approaches. We also evaluate SSDH on a large dataset containing more than 1 million images to demonstrate its scalability.

To sum up, the main contributions of this paper include:

- Unifying retrieval and classification: We propose a supervised hashing approach, SSDH, that takes advantage of deep learning, unifies classification and retrieval in a single learning model, and jointly learns representations, hash functions, and a linear classifier from image data.
- Scalable hash: Our SSDH performs learning in a pointwised manner, and thereby requires neither pairs nor triplets of image inputs. This characteristic makes it easily scalable to large-scale image retrieval.
- Lightweight hash: SSDH leverages the effective deep architecture for classification and can be easily realized by a slight modification of an existing deep classification network.
- Large-scale data validation: We conduct extensive exper-

H.-F. Yang and C.-S. Chen are with the Research Center for Information Technology Innovation, Academia Sinica, Taipei, Taiwan (e-mail: hfyang@citi.sinica.edu.tw; song@iis.sinica.edu.tw).

K. Lin and C.-S. Chen are with the Institute of Information Science, Academia Sinica, Taipei, Taiwan (e-mail: kevinlin311.tw@iis.sinica.edu.tw).

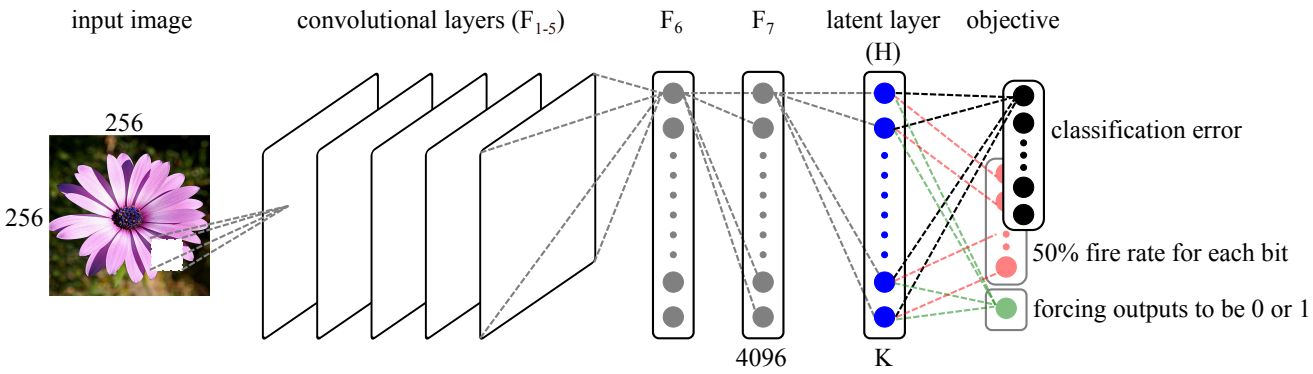

Fig. 1. An overview of our proposed supervised semantic-preserving deep hashing (SSDH). We construct the hash functions as a latent layer with K units between the image representation layer and classification outputs in a convolutional neural network (CNN) originally comprising 5 convolutional layers, 2 fully connected layers, and an output layer. SSDH takes inputs from images of a size of 256×256 pixels and learns image representations, binary codes, and classification through the optimization of an objective function that combines a classification loss with desirable properties of hash codes. The learned codes preserve the semantic similarity between images and are compact for image search.

iments on several benchmarks and one large collection of more than 1 million images and show that SSDH produces promising results.

Preliminary results of this work have been published in [8], [9], and the contributions are presented here as a whole. The extensions in this work include the following: (1) A new objective is introduced that incorporates additional constraints to enable learning approximately 0-1 valued, compact and balanced codes; (2) more in-depth experiments have been done to evaluate the performance on various datasets with comparisons to state-of-the-art methods, including experiments on a large-scale dataset that are difficult to handle by many non-scalable methods; and (3) more rationals/analyses and discussions have been added for the architecture design and experimental results, respectively.

II. Background

Learning-based hashing algorithms construct hash codes by leveraging the training data and are expected to overcome the limitations of data-independent methods such as methods [5]–[7], [10] in the LSH family. The learning-based approaches can be grouped into three categories according to the degree of supervised information of labeled data used: unsupervised, semi-supervised, and supervised methods. Unsupervised algorithms [1], [3], [11], [12] use unlabeld data for code construction and try to preserve the similarity between data examples in the original space (e.g., the Euclidean space). Representative methods in this category include spectral hashing (SH) [12], kernelized locality-sensitive hashing (KLSH) [3], and iterative quantization (ITQ) [1].

Semi-supervised algorithms [13]–[15] use information from both the labeled samples and the unlabeled ones for learning hash functions. For example, the semi-supervised hashing (SSH) [14] minimizes the empirical error on the pairwise labeled data (e.g., similar and dissimilar data pairs) and maximizes the variance of hash codes over the labeled and unlabled data. The semi-supervised tag hashing (SSTH) [15] models the correlation between the hash codes and the class labels in a supervised manner and preserves the similarity between image examples in an unsupervised manner.

Supervised hashing approaches [16]–[24] aim to fully take advantage of the supervised information of labeled data for learning more efficient binary representations so as to attain higher search accuracy than the unsupervised and the semi-supervised approaches. Utilizing pairwise relations between data samples, binary reconstructive embedding (BRE) [16] minimizes the squared error between the original Euclidean distances and the Hamming distances of binary codes. Minimal loss hashing (MLH) [20] minimizes the empirical loss for code construction. Ranking-based methods [21], [24] that leverage the ranking information from a set of triplets have also been proposed. Methods that rely on pairs or triplets of image samples for training generally need a high storage cost and are infeasible for large datasets. Learning binary codes in a pointwised manner would be a better alternative for the scalability of hash. Pointwise methods use the provided label information to guide the learning of hash functions. Boosted similarity sensitive coding (BCCS) [25] is one such method that learns a weighted Hamming embedding by exploiting the labels in a boosting algorithm (AdaBoost). We have been inspired to model the relationship between the semantic labels and the binary codes in the hash function learning, under the assumption that the semantic labels are represented by binary concepts. In parallel with our development, the supervised discrete hashing (SDH) [23] formulates the learning of hash codes in terms of classification in order to learn binary codes optimal for classification. While SDH and ours share similar spirits on designing hash codes, SDH decomposes the hashing learning into sub-problems and needs a carefully chosen loss function for classification to make the entire optimization efficient and scalable. Our formulation on the deep networks simplifies the optimization process and is naturally scalable to large-scale datasets.

In the learning-based hashing approaches, methods based on deep networks [26]–[32] form a special group so we discuss them separately here. One of the earliest efforts to apply deep networks in hashing is semantic hashing (SH) [30]. It constructs hash codes from raw images via a network with stacked Restricted Boltzmann Machines (RBMs). The learned binary codes are treated as memory addresses, and

thus finding similar items to the query can be done by simply accessing to memory addresses that are within a Hamming ball around the query vector. Deep networks are also used in deep hashing (DH) and supervised DH (SDH) [28] for learning compact binary ocdes through seeking multiple non-linear projections to map samples into binary codes, together with optimizing several criteria of binary codes. Recently, hashing methods based on CNNs have also been proposed. CNNH and CNNH+ [31] employ a two-stage learning approach that first decomposes a pairwise similarity matrix into approximate hash codes and then trains a CNN for learning the hash functions. The method in [27] and deep semantic ranking based hashing (DSRH) [32] adopt a ranking loss defined on a set of triplets for code construction. Although these deep learning-based hashing approaches have produced promising results, some need to perform a matrix factorization prior to the hash function learning (e.g., CNNH and CNNH+ [31]) and some need to take inputs in the form of image pairs (e.g., SDH [28]) or image triples (e.g., [27] and DSRH [32]), which make them less favorable in the case when the data size is large.

III. Learning Hash Codes via Deep Networks

Let $\mathcal{I} = \{I_n\}_{n=1}^N$ be the N images and $\mathcal{Y} = \{y_n\}^{M\times N}$ be their associated label vectors, where M denotes the total number of class labels. An entry of y_n is 1 if an image I_n belongs to the corresponding class and 0 otherwise. Our goal is to learn a mapping $\mathcal{F} : \mathcal{I} \rightarrow \{0,1\}^{K\times N}$, which maps images to their k-bit binary codes $B = \{b_n\} \in \{0,1\}^{K\times N}$ while preserving the semantic similarity between image data. Specifically, we aim to design a supervised hashing algorithm that exploits the semantic labels to automatically create binary codes of the following properties:

- The codes respect the semantic similarity between image labels. Images that share common class labels are mapped to same (or similar) binary codes.
- The codes are evenly distributed and discriminative. Each binary bit carries as much information as possible.

A. Deep Hashing Functions

We take advantage of recent advances in deep learning and construct the hash functions on a CNN that is capable of learning semantic representations from images. Studies have shown that the learned deep features capture rich image representations and provide better performance than the handcrafted features on image classification [33], object detection [34], [35], semantic segmentation [34], [36] and image retrieval [31], [32].

Our network is based on the deep model of AlexNet [37], which has 5 convolutional layers (F_{1-5}) with max-pooling operations, followed by 2 fully connected layers (F_{6-7}) and an output layer. The hidden layers (F_{1-7}) in AlexNet are composed of the rectified linear units (ReLUs) because the ReLUs lead to faster training than other non-linearities such as the hyperbolic tangent and logistic sigmoid [38]. The output units are with the softmax functions and the network is trained to maximize the multinomial logistic regression objective function for multi-class classification. To incorporate the deep representations into the hash function learning, we add a *latent layer* H with K units to the top of layer F_7, as illustrated in Figure 1. This latent layer is fully connected to F_7 and uses the sigmoid units so that the activations are bounded between 0 and 1.

Let $W^H \in \mathbb{R}^{d\times K}$ denote the weights (i.e. the projection matrix) in the latent layer. For a given image I_n with the feature vector $a_n^7 \in \mathbb{R}^d$ in layer F_7, the activations of the units in H can be computed as $a_n^H = \sigma(a_n^7 W^H + e^H)$, where e^H is the bias term and $\sigma(\cdot)$ the logistic sigmoid function, defined by $\sigma(z) = 1/(1+\exp(-z))$, with z a real value. The binary encoding function is given by

$$b_n = \text{sign}(\sigma(a_n^7 W^H + e^H) - 0.5) = \text{sign}(a_n^H - 0.5), \quad (1)$$

where $\text{sign}(v) = 1$ if $v > 0$ and 0 otherwise, and $\text{sign}(\cdot)$ performs element-wise operations for a matrix or a vector.

B. Label Consistent Binary Codes

Image labels not only provide knowledge in classifying images but also are useful supervised information for learning hash functions. We propose to model the relationship between the labels and the binary codes in order to construct semantics-preserving binary codes. We assume that the semantic labels can be derived from a set of K latent concepts (or latent attributes) with each attribute being *on* or *off*. When an input image is associated with binary-valued outputs (in $\{0,1\}^K$), the classification is dependent on these hidden attributes. This implies that through an optimization of a loss function defined on the classification error, we can ensure that semantically similar images are mapped to similar binary codes.

Consider a matrix $W^C \in \mathbb{R}^{K\times M}$ that performs a linear mapping of the binary hidden attributes to the class labels. Incorporating such a matrix into our the network amounts to adding a classification layer to the top of the latent layer (see Figure 1 where the black dashed lines denote W^C). Let $\hat{y}_n$ denote the prediction of our network (the black nodes in Figure 1) for an image I_n. In terms of the classification formulation, one can choose to optimize the following objective function:

$$\arg\min_W \quad \sum_{n=1}^N L(y_n, \hat{y}_n) + \lambda ||W||^2, \quad (2)$$

where $L(\cdot)$ is a loss function that minimizes classification error and will be detailed below, W denotes the weights of the network, and λ governs the relative importance of the regularization term.

C. Efficient Binary Codes

In addition to similarity preserving, the binary codes should be compact and discriminative, which requires that each bit fires 50% of the time [12]. This can be achieved by minimizing $\sum_{n=1}^N (\text{mean}(a_n^H) - 0.5)^2$, which in some sense is equivalent to maximizing the entropy of a binary vector. However, optimizing this criterion alone may lead to extreme cases where the activation of every node in the latent layer is around

0.5. We thus enforce another criterion that encourages the activation of each node in H to be close to 0 or 1 as much as possible. Hence, to make the binary codes efficient, we aim to optimize the following objective:

$$\arg\min_{W} \quad -\sum_{n=1}^{N}||a_n^H - 0.5||^2 + \sum_{n=1}^{N}(\text{mean}(a_n^H) - 0.5)^2, \quad (3)$$

where the first term maximizes the difference between the activation of a unit and the optimal mean, encouraging the activations of the units in H to be close to either 0 or 1, and the second minimizes the difference between the mean activation of a_n^H and the optimal mean, ensuring that the output of each node has a 50% chance of being 0 or 1.

On the network design, we add a unit (the green node in Figure 1) that performs an average pooling operation (the green dashed lines) over the nodes in the latent layer to obtain the mean activation for the optimization of the second term in Equation (3). The weights associated with the connections to this unit are fixed to $1/K$. The first term in Equation (3) imposes constraints directly on the units in the latent layer. No modification to the network is needed. However, for the clarity of presentation, we draw additional red nodes in Figure 1 to indicate this constraint.

D. Overall Objective

The entire objective function aiming for constructing similarity preserving (Equation (2)) and efficient (Equation (3)) binary codes is given as:

$$\begin{aligned}\arg\min_{W} \quad & \alpha\sum_{n=1}^{N}L(y_n, \hat{y}_n) + \lambda||W||^2 \\ & - \beta\sum_{n=1}^{N}||a_n^H - 0.5||^2 \\ & + \gamma\sum_{n=1}^{N}(\text{mean}(a_n^H) - 0.5)^2, \end{aligned} \quad (4)$$

where α, β, and γ are parameters that control the weighting of each term, which are set to 1 in our experiments.

We implement our approach by using the open source CAFFE [39] package. As our network is adopted from AlexNet [37] that has been trained on the 1.2 million ILSVRC subset of the ImageNet for the 1000-class recognition task, the initial weights in layers F_{1-7} of our network are set as the pre-trained ones and the remaining weights are randomly initialized. Stochastic gradient descent (SGD), in conjunction with backpropagation, is applied to network training, with a mini-batch size of 32 images for minimizing the overall objective defined in Equation (4). We also employ dropout in which the activations of the hidden units are set to zero with a probability of 0.5 during training in order to avoid over-fitting. Our model is a lightweight modification of AlexNet and thus is easy to implement. The codes are publicly available[1].

[1]https://gist.github.com/kevinlin311tw

IV. Experiments

We conduct experiments on three benchmarks and one large dataset embracing more than 1 million images. The images in three benchmarks are in a wide spectrum of image types including tiny objects of CIFAR-10, handwritten digits of MNIST, and scene images of SUN397. The large dataset, Yahoo-1M, comprises product images with heterogeneous types and is used to demonstrate that our SSDH is easily scaled up to large-scale image data.

In the following, we first introduce the evaluation protocols, and then describe the datasets. Finally, experimental results and extensive evaluation are presented.

A. Evaluation Protocols

We use three evaluation metrics widely used in the literature for the performance comparison. These metrics measure the performance of hashing algorithms from different aspects.

- Mean average precision (mAP) at k samples: We rank all the images according to their Hamming distances to the query image, select a set of k images from the top of the ranked list as retrieval results, and compute mAP on these returned images. We set k as $1,000$ in the experiments. The mAP computes the area under the recall-precision curve and is an indicator of the overall performance of hash functions;
- Precision at k samples: It is computed as the percentage of true neighbors among the top k retrieved images;
- Precision within Hamming radius r: We compute the precision of the images in the buckets that fall within the Hamming radius 2 of the query image. A failed search that returns no images receives zero precision.

We use the class labels as the ground truth and follow the common settings that all the above three metrics are computed through examining whether the returned images and the query share common class labels.

B. Datasets

CIFAR-10: The CIFAR-10[2] [40] dataset consists of $60,000$ 32×32 color images categorized into 10 classes. The class labels are mutually exclusive, and thus each class has $6,000$ images. The entire dataset is partitioned into two non-overlapping sets: a training set with $50,000$ images and a test set with $10,000$ images.

MNIST: The MNIST[3] dataset is a collection of $70,000$ 28×28 grayscale images of handwritten digits grouped into 10 classes. The entire dataset comprises of a training set of $60,000$ images and a test set of $10,000$ images.

SUN397: The SUN397[4] [41] dataset comprises $108,754$ scene images in 397 categories. The number of images varies across categories, with each category containing at least 100 images. Following the settings in [18], we select randomly $8,000$ images to form the test set and use the remaining $100,754$ as training samples.

[2]http://www.cs.toronto.edu/ kriz/cifar.html
[3]http://yann.lecun.com/exdb/mnist/
[4]http://groups.csail.mit.edu/vision/SUN/

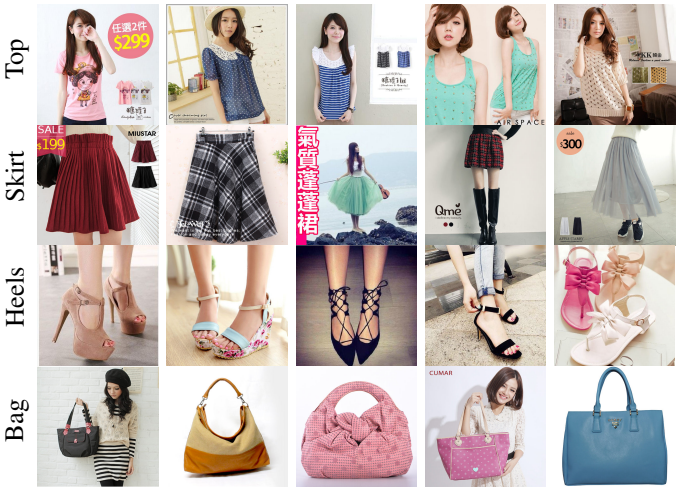

Fig. 2. Selected sample images from the Yahoo-1M shopping dataset. The dataset contains product images of heterogeneous types, including those that are backgroundless or of cluttered backgrounds, with or without humans.

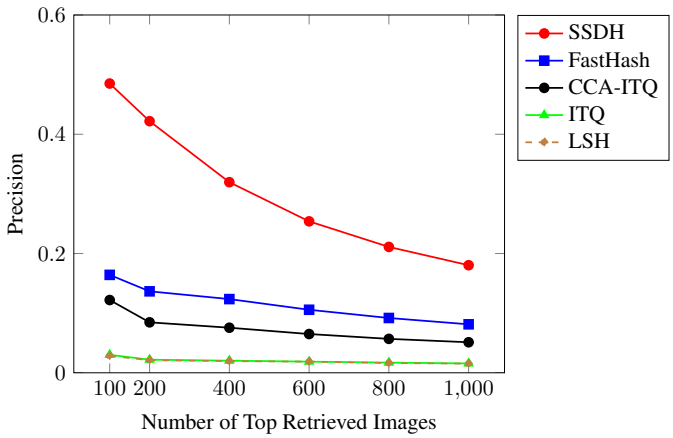

Fig. 5. Precision curves with respect to different number of top retrieved samples on the SUN397 dataset when the 1024-bit hash codes are used in the evaluation.

Yahoo-1M Shopping Images: The Yahoo-1M dataset embraces $1,124,086$ product images of heterogeneous types collected from the Yahoo shopping sites: the images are of cluttered backgrounds or backgroundless, with or without humans. Figure 2 shows some selected images. Each image is associated with a class label, and there are 116 classes in total. The number of images in each class varies greatly, ranging from $1,007$ to $150,211$. To divide the dataset into two sets, each consisting of the images from each class, we selected 90% of the images from each class as training samples and the rest 10% as test samples. The entire dataset is thus partitioned into a training set of $1,011,723$ images and a test set of $112,363$ images.

Note that our network takes fixed-size image inputs; hence, all the training and test samples, regardless of their original sizes, were resized to 256×256 prior to training or testing.

C. Results on CIFAR-10

We compared our approach with several hashing methods, including unsupervised methods (LSH [5], ITQ [1], and SH [12]) and supervised approaches (BRE [16], MLH [20], CCA-ITQ [1], CNNH+ [31], CNNH [31], and Lai et al. [27]). Among the six supervised approaches, CNNH+, CNNH, and Lai et al., like our approach, take advantage of deep learning techniques for constructing compact binary codes.

Figure 3a shows the comparative results based on the mAP of top $1,000$ returned images as a function of code length. As can be seen, SSDH provides the best performance for different code lengths. Moreover, it improves the mAP by a margin of around 28% compared with the method proposed by Lai et al. [27]. These results suggest that unifying retrieval and classification in a single learning model in which the hash code learning is governed by the semantic labels can better capture the semantic structure between images and hence yields better performance. Another observations are that (1) the supervised approaches constantly perform better than the unsupervised ones and that (2) among all the supervised approaches, the deep learning-based approaches give relatively better results, and this can be attributed to the fact that deep networks enable joint learning of representations and hash functions from images and the learned representations are more effective than the hand-engineered ones.

Figure 3b shows the precision at k samples, where k ranges from 100 to $1,000$, when the 48-bit hash codes are used in the evaluation. These curves convey similar messages as observed in Figure 3a. Our approach has a consistent advantage over other hashing methods, and the approaches (ours, Lai et al., CNNH+, CNNH, and CCA-ITQ) that exploit the label information in learning hash functions perform better than those that do not.

The evaluation of the precision within Hamming radius 2 is shown in Figure 3c. Our approach performs favorably against other approaches although the performance gains become smaller when the code length increases. This is because our approach needs more nodes in the latent layer for learning longer hash codes, resulting in substantially increased number of weights that need to be learned. However, the CIFAR-10 provides relatively small amount of data (only $5,000$ images per class) for training. When the samples are not sufficient for training, a deep network is prone to over-fitting.

D. Results on MNIST

Figure 4 shows the performance comparison of different hashing methods on MNIST. We see that these results accord with our observations in CIFAR-10 that SSDH performs favorably against other supervised and unsupervised approaches and that the hashing approaches based on deep models achieve better performance than the conventional methods that map the hand-engineered features to the binary codes.

E. Results on SUN397

SUN397 comprises more $100,000$ images and 397 scene categories, it is more challenging than CIFAR-10 and MNIST, and it may need longer code lengths for hashing methods to achieve good performance. Figure 5 compares SSDH,

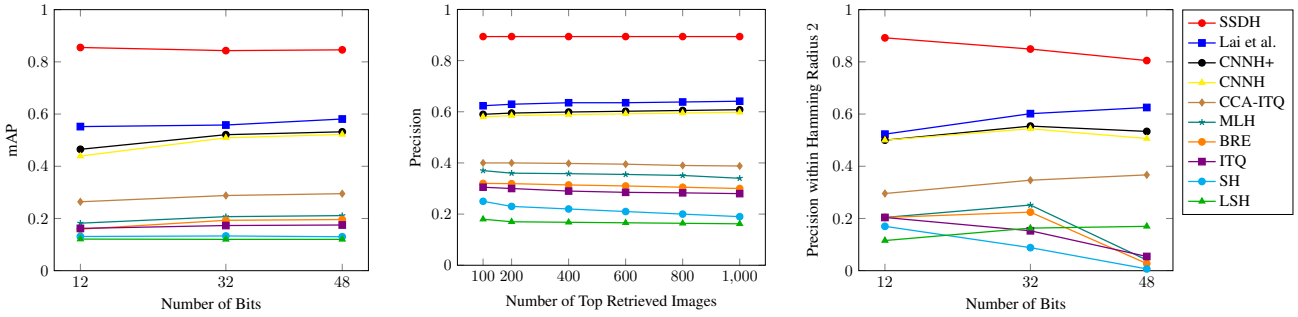

Fig. 3. Comparative evaluation of different hashing algorithms on the CIFAR-10 dataset. (a) mAP curves with respect to different number of hash bits. (b) Precision curves with respect to different number of top retrieved samples when the 48-bit hash codes are used in the evaluation. (c) Precision within Hamming radius 2 curves with respect to different number of hash bits. Our approach provides the best performance when evaluated by these three metrics.

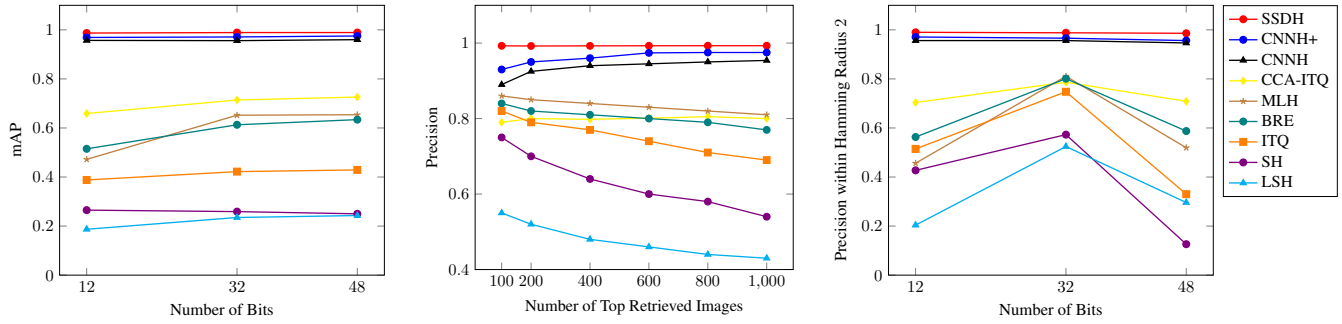

Fig. 4. Comparative evaluation of different hashing algorithms on the MNIST dataset. (a) mAP curves with respect to different number of hash bits. (b) Precision curves with respect to different number of top retrieved samples when the 48-bit hash codes are used in the evaluation. (c) Precision within Hamming radius 2 curves with respect to different number of hash bits. Our approach provides the best performance when evaluated by these three metrics.

TABLE I
MAP OF VARIOUS METHODS AT 128 BITS ON THE YAHOO-1M DATASET. ALEXNET-FT DENOTES THAT THE FEATURES FROM LAYER F_7 OF ALEXNET FINE-TUNED ON YAHOO-1M ARE USED IN LEARNING HASH CODES.

Method	mAP
l_2 + AlexNet-ft	0.4895
LSH + AlexNet-ft	0.4639
ITQ + AlexNet-ft	0.5386
CCA-ITQ + AlexNet-ft	0.6169
SSDH	0.6470

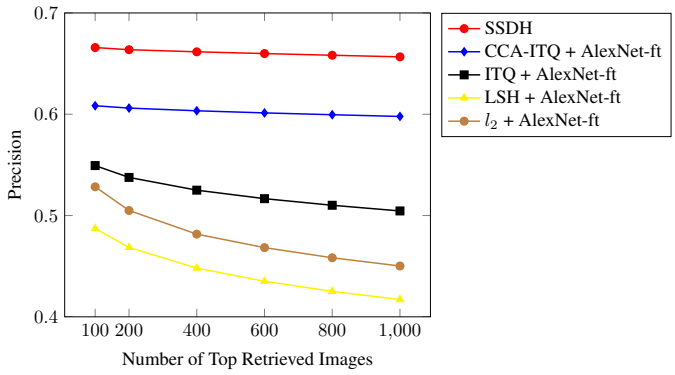

Fig. 6. Precision curves with respect to different number of top retrieved samples on the Yahoo-1M dataset when the 128-bit hash codes are used in the evaluation. AlexNet-ft denotes that the features from layer F_7 of AlexNet fine-tuned on Yahoo-1M are used in learning hash codes.

FastHash [18], CCA-ITQ, ITQ, and LSH based on the precision at different number of top returned images when the 1024-bit hash codes are used. SSDH performs better than other approaches regardless of the number of top returned images. Moreover, it has more advantages when a small number of top returned images are needed. When only the top 200 returned images are considered, SSDH outperforms FastHash by a margin of 30% precision. Thus, even for the case with long code length, SSDH achieves state-of-the-art retrieval performance.

F. Results on Yahoo-1M Dataset

Deep features have been proven to be more effective than handcrafted ones. We thus fine-tuned AlexNet on Yahoo-1M and used the features from layer F_7 of the fine-tuned network for LSH, ITQ, and CCA-ITQ to learn hash codes. To provide more insight into the performance of the hashing approaches, we also included the results obtained by an exhaustive search based on the Euclidean (l_2) distance of the deep features from

TABLE II
CLASSIFICATION PERFORMANCE OF VARIOUS METHODS ON CIFAR-10. THE PERFORMANCE IS MEASURED BY THE TEST ERROR RATE.

Method	Test Error (%)
Stochastic Pooling [42]	15.13
CNN + Spearmint [43]	14.98
AlexNet + Fine-tuning	11.69
NIN + Dropout [44]	10.41
NIN + Dropout + Augmentation [44]	8.81
SSDH w/ 12-bit codes	10.69
SSDH w/ 32-bit codes	10.66
SSDH w/ 48-bit codes	10.47
SSDH w/ 64-bit codes	10.25

TABLE III
CLASSIFICATION PERFORMANCE OF VARIOUS METHODS ON MNIST. THE PERFORMANCE IS MEASURED BY THE TEST ERROR RATE.

Method	Test Error (%)
Stochastic Pooling [42]	0.47
AlexNet + Fine-tuning	0.84
NIN + Dropout [44]	0.47
SSDH w/ 12-bit codes	0.75
SSDH w/ 32-bit codes	0.72
SSDH w/ 48-bit codes	0.78
SSDH w/ 64-bit codes	0.62

the fine-tuned AlexNet in the comparison. In this experiment, the performance of the hashing approaches is evaluated when the code length is 128.

Table I shows the mAP of the top $1,000$ returned images and Figure 6 the precision curves with respect to different number of top retrieved images. Surprisingly, all the hashing approaches, except LSH, give better retrieval performance than a direct comparison based on the Euclidean distance of the deep features. This shows that learning hash codes on the deep features improves the retrieval performance, and supervised hashing approaches can better capture the semantic structure of the data than unsupervised ones. Furthermore, the traditional hashing methods learn the mapping of features to binary codes, but not the features. Our SSDH, in contrast, allows for simultaneous learning of the deep features and the hash functions and hence gives the best retrieval results.

G. Image Classification

As our SSDH unifies retrieval and classification in a single learning process, it can be applied to image classification problems. We used CIFAR-10 and MNIST and compared SSDH with other state-of-the-arts on the classification task. The classification accuracy of various approaches is reported in Tables II and III. On CIFAR, as can be seen from Table II, SSDH with different hash code lengths attain higher classification accuracy than several approaches in which the fine-tuned AlexNet is served as a baseline for our approach, and yields comparable performance to the Network In Network (NIN) [44]. With regard to the results of MNIST in Table III, SSDH has an advantage over the fine-tuned AlexNet but is slightly inferior to the others. These results suggest that unifying retrieval and classification in a single model not only produces promising classification results when compared with the models that are optimized for a classification task, but also is beneficial to the retrieval task as shown in Sections IV-C to IV-F.

V. CONCLUSIONS

We have presented a supervised deep hashing model, SSDH, that preserves the label semantics between images. SSDH constructs hash functions as a latent layer between the feature layer and the classification layer in a network. By optimizing an objective function defined over classification error and desired criterion for binary codes, SSDH jointly learn binary codes, features, and classification. Such a network design comes with several merits: (1) SSDH unifies retrieval and classification in a single model; and (2) SSDH is simple and is easily realized by a slight modification of an existing deep network for classification; and more importantly (3) SSDH is naturally scalable to large scale search. We have conducted extensive experiments and have provided comparative evaluation of SSDH with several state-of-the-arts on three benchmarks with a wide range of image types and one large dataset including more than 1 million images. The results have shown that SSDH achieves superior retrieval performance and provides promising classification results.

REFERENCES

[1] 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.*, vol. 35, no. 12, pp. 2916–2929, 2013.

[2] J. He, W. Liu, and S. Chang, "Scalable similarity search with optimized kernel hashing," in *Proc. ACM SIGKDD*, 2010, pp. 1129–1138.

[3] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 6, pp. 1092–1104, 2012.

[4] X. Liu, J. He, B. Lang, and S. Chang, "Hash bit selection: A unified solution for selection problems in hashing," in *Proc. CVPR*, 2013, pp. 1570–1577.

[5] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," in *Proc. FOCS*, 2006, pp. 459–468.

[6] M. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proc. ACM STOC*, 2002, pp. 380–388.

[7] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proc. VLDB*, 1999, pp. 518–529.

[8] K. Lin, H.-F. Yang, J.-H. Hsiao, and C.-S. Chen, "Deep learning of binary hash codes for fast image retrieval," in *Proc. CVPRW on DeepVision: Deep Learning in Computer Vision*, 2015.

[9] K. Lin, H.-F. Yang, K.-H. Liu, J.-H. Hsiao, and C.-S. Chen, "Rapid clothing retrieval via deep learning of binary codes and hierarchical search," in *Proc. ICMR*, 2015.

[10] M. Raginsky and S. Lazebnik, "Locality-sensitive binary codes from shift-invariant kernels," in *Proc. NIPS*, 2009, pp. 1509–1517.

[11] W. Liu, J. Wang, S. Kumar, and S. Chang, "Hashing with graphs," in *Proc. ICML*, 2011, pp. 1–8.

[12] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. NIPS*, 2008, pp. 1753–1760.

[13] Y. Mu, J. Shen, and S. Yan, "Weakly-supervised hashing in kernel space," in *Proc. CVPR*, 2010, pp. 3344–3351.

[14] J. Wang, S. Kumar, and S. Chang, "Semi-supervised hashing for large-scale search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 12, pp. 2393–2406, 2012.

[15] 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 *Proc. ECCV*, 2014, pp. 378–392.

[16] B. Kulis and T. Darrell, "Learning to hash with binary reconstructive embeddings," in *Proc. NIPS*, 2009, pp. 1042–1050.

[17] G. Lin, C. Shen, D. Suter, and A. van den Hengel, "A general two-step approach to learning-based hashing," in *Proc. ICCV*, 2013, pp. 2552–2559.

[18] G. Lin, C. Shen, Q. Shi, A. van den Hengel, and D. Suter, "Fast supervised hashing with decision trees for high-dimensional data," in *Proc. CVPR*, 2014, pp. 1971–1978.

[19] W. Liu, J. Wang, R. Ji, Y. Jiang, and S. Chang, "Supervised hashing with kernels," in *Proc. CVPR*, 2012, pp. 2074–2081.

[20] M. Norouzi and D. J. Fleet, "Minimal loss hashing for compact binary codes," in *Proc. ICML*, 2011, pp. 353–360.

[21] M. Norouzi, D. J. Fleet, and R. Salakhutdinov, "Hamming distance metric learning," in *Proc. NIPS*, 2012, pp. 1070–1078.

[22] G. Shakhnarovich, P. A. Viola, and T. Darrell, "Fast pose estimation with parameter-sensitive hashing," in *Proc. ICCV*, 2003, pp. 750–759.

[23] F. Shen, C. Shen, W. Liu, and H. T. Shen, "Supervised discrete hashing," in *Proc. CVPR*, 2015.

[24] J. Wang, W. Liu, A. X. Sun, and Y. Jiang, "Learning hash codes with listwise supervision," in *Proc. ICCV*, 2013, pp. 3032–3039.

[25] G. Shakhnarovich, "Learning task-specific similarity," Ph.D. dissertation, Massachusetts Institute of Technology, 2005.

[26] A. Krizhevsky and G. E. Hinton, "Using very deep autoencoders for content-based image retrieval," in *Proc. ESANN*, 2011.

[27] H. Lai, Y. Pan, Y. Liu, and S. Yan, "Simultaneous feature learning and hash coding with deep neural networks," in *Proc. CVPR*, 2015.

[28] V. E. Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou, "Deep hashing for compact binary codes learning," in *Proc. CVPR*, 2015.

[29] A. Torralba, R. Fergus, and Y. Weiss, "Small codes and large image databases for recognition," in *Proc. CVPR*, 2008.

[30] R. Salakhutdinov and G. E. Hinton, "Semantic hashing," *Int. J. Approx. Reasoning*, vol. 50, no. 7, pp. 969–978, 2009.

[31] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, "Supervised hashing for image retreieval via image representation learning," in *Proc. AAAI*, 2014.

[32] F. Zhao, Y. Huang, L. Wang, and T. Tan, "Deep semantic ranking based hashash for multi-label image retreieval," in *Proc. CVPR*, 2015.

[33] D. C. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proc. CVPR*, 2012.

[34] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. CVPR*, 2014.

[35] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," in *Proc. ICLR*, 2014.

[36] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1915–1929, 2013. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/TPAMI.2012.231

[37] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012.

[38] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. AISTATS*, vol. 15, 2011, pp. 315–323.

[39] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. B. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. ACM MM*, 2014, pp. 675–678.

[40] A. Krizhevsky, "Learning multiple layers of features from tiny images," *Computer Science Department, University of Toronto, Tech. Report*, 2009.

[41] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "SUN database: Large-scale scene recognition from abbey to zoo," in *Proc. CVPR*, 2010, pp. 3485–3492.

[42] M. D. Zeiler and R. Fergus, "Stochastic pooling for regularization of deep convolutional neural networks," in *Proc. ICLR*, 2013.

[43] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Proc. NIPS*, 2012, pp. 2960–2968.

[44] M. Lin, Q. Chen, and S. Yan, "Network in network," in *Proc. ICLR*, 2014.