Towards Category Based Large-Scale Image Retrieval Using Transductive Support Vector Machines

Hakan Cevikalp[1(✉)], Merve Elmas[1], and Savas Ozkan[2]

[1] Electrical and Electronics Engineering,
Eskisehir Osmangazi University, Eskişehir, Turkey
hakan.cevikalp@gmail.com, merveelmas1@gmail.com
[2] TUBITAK UZAY, Ankara, Turkey
savas.ozkan@tubitak.gov.tr

Abstract. In this study, we use transductive learning and binary hierarchical trees to create compact binary hashing codes for large-scale image retrieval applications. We create multiple hierarchical trees based on the separability of the visual object classes by random selection, and the transductive support vector machine (TSVM) classifier is used to separate both the labeled and unlabeled data samples at each node of the binary hierarchical trees (BHTs). Then the separating hyperplanes returned by TSVM are used to create binary codes. We propose a novel TSVM method that is more robust to the noisy labels by interchanging the classical Hinge loss with the robust Ramp loss. Stochastic gradient based solver is used to learn TSVM classifier to ensure that the method scales well with large-scale data sets. The proposed method improves the Euclidean distance metric and achieves comparable results to the state-of-art on CIFAR10 and MNIST data sets and significantly outperforms the state-of-art hashing methods on NUS-WIDE dataset.

Keywords: Image retrieval · Transductive support vector machines · Semi-supervised learning · Ramp loss

1 Introduction

Large-scale image retrieval has recently attracted great attention due to the rapid growth of visual data brought by Internet. Image retrieval can be defined as follows: Given a query image, finding and representing (in an ordered manner) the images depicting the same scene or objects in large unordered image collections. Despite the great research efforts, image retrieval is still a challenging problem since large-scale image search demands highly efficient and accurate retrieval methods.

For large scale image search, the most commonly used method is the hashing method that enables us to approximate the nearest neighbor search. Hashing methods convert each image feature vector in the database into a compact code

G. Hua and H. Jégou (Eds.): ECCV 2016 Workshops, Part I, LNCS 9913, pp. 621–637, 2016.
DOI: 10.1007/978-3-319-46604-0_44

(typically a binary code) and provide constant or sub-linear search time. Most of the current popular hashing methods [6,7,10,17,18,22,26,27,31] are unsupervised methods and they are built on the assumption that the similar images in the Euclidean space must have similar binary codes. Among these, Locality Sensitive Hashing (LSH) [6] chooses random projections so that two closest image samples in the feature space falls into the same bucket with a high probability. However, due to the semantic gap between the low-level features and semantics, Euclidean distances in the feature space do not reflect the semantic similarities between the images. Furthermore, the state-of-art image visual features are typically high-dimensional vectors ranging from several thousands to millions. As pointed out in [1], the performance of nearest-neighbor techniques using the Euclidean distances in high-dimensional spaces is poor since sparse and irregular distributions of data samples tend to have many holes (regions that have few or no nearby samples from the same classes), so it is necessary to learn more discriminative distance metrics. Therefore, relying Euclidean distances between image feature vectors for creating binary hash codes can be misleading. Our experimental results at the end also verify these claims.

To solve the challenging semantic gap problem, the most straightforward solution is to use label information. But, labeling all images in large image databases is too costly and difficult in practice. In contrast, relevant feedback given in terms of similar/dissimilar pairs is much easier to collect. Similarly, for most images on the web, some label tags can be collected at a more reasonable cost by using image file names or surrounding text. So, both semi-supervised and supervised hashing methods utilizing these types of information have been proposed [9,11,15,21,23,24,30,34]. Majority of these methods [9,21,24,30,34] use label information during creating similarity matrix and then projection directions that will preserve the similarities within the similarity matrix are found. Finally, these directions are used to produce binary codes. These methods cannot be applied directly to large-scale image datasets since they require computing and operating on a very large $n \times n$ sized similarity matrix, where n is the total number of image samples in the training (gallery) set. Generally, two procedures are followed to avoid this problem: In the first approach, only a small number of labeled data samples is used to learn binary codes and all unlabeled data samples are ignored. In the second procedure, some representative anchor points are created by random selection or clustering, and the similarity matrix of all data is approximated with much smaller sized similarity matrix of those anchor points. Both procedures are problematic in the sense that some potential information that may come from unlabeled data samples are ignored and propagation of supervised information from labeled samples to neighboring unlabeled samples has not been taken into consideration. The methods [11,15,23] that are more related to ours use SVM based large margin classifiers to learn compact binary codes. Both [11] and [23] use only labeled data since their methods require to operate on $n \times n$ sized kernel matrix. Thus, unlabeled data are ignored again and they do not contribute to label propagation. [15] does not need any supervision and the authors randomly select some samples and randomly assign them posi-

tive and negative labels. Then they run SVM algorithm to find the hyperplanes separating these samples and finally separating hyperplanes are used to produce binary codes. More recently, deep neural networks and CNN (Convolutional Neural Networks) features have been used for image retrieval [8,19,20,32,35–37]. These methods typically follow the similar structure of classifier networks that use a stack of convolutional layers to produce discriminative features, but the last layers use different loss functions that are more suitable for retrieval applications. For example, [19,35] use a triplet ranking loss designed to characterize one image is more similar to the second image than the third one whereas some methods use other loss functions such as surrogate loss [37], pair-wise ranking [8], or weighted approximate ranking [8]. Almost all these deep neural network methods are trained end-to-end fashion, which allows one to optimize the discriminative image features and hash functions simultaneously.

Similar to the hashing methods using large-margin based classifiers, we also use SVM classifiers to learn binary codes, but in contrast to other methods, we incorporate the unlabeled data during learning process in a transductive learning setting. Since the labeled data can be noisy in large-scale image retrieval applications, we introduce a more robust transductive SVM (TSVM) method to the noise present in labels. We use stochastic gradient based solver instead of sequential minimal optimization (SMO), thus our method scales well with large-scale data (to the best of our knowledge, it is the only transductive method that can be used with more than a million data). Finally, we introduce a novel method to learn class hierarchies based on graph cut and binary hierarchical trees to ensure the large margin between class samples.

2 Method

Here we consider the scenario where we have many unlabeled images with some limited amount of labeled image data. As we mentioned earlier, labels can be gathered from image file names or nearby text on the web. But, we have to keep in mind that the labels can be very noisy and there might be more than one labels attached to an image, e.g., if an image contains people, car, buildings etc., all these tags can be used to label the image. We use TSVMs to create binary hash codes. It should be noted that SVM like large-margin based classifiers are widely used for this goal and it was shown that larger margin between class samples yields to lower error rates in similarity search [15,23]. So, our goal is to find separating hyperplanes which will create balanced binary hash codes but at the same time they will yield to a large margin between the image samples of different classes.

In the proposed methodology, we first create image class hierarchies based on their visual content similarities and labels. To this end, we use binary hierarchical trees and Normalized Cuts clustering. This methodology is much better compared to the Wordnet based hierarchy used in [5] since it is created based on the separability of the visual classes. Then, we use TSVM to find the hyperlane that best separates the data samples (both labeled and unlabeled data) at each

node of the binary hierarchical tree to make sure that the classes are split into two clusters with the largest margin possible. We first explain our novel TSVM algorithm and then describe how to create binary hierarchical trees and compact binary codes below.

2.1 Robust Transductive Support Vector Machines (RTSVMs)

Suppose that we are given a set of L labeled training samples $L = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_L, y_L)\}$, $\mathbf{x} \in \mathbb{R}^d$, $y \in \{+1, -1\}$ and an unlabeled set of U vectors $U = \{\mathbf{x}_{L+1}, \ldots, \mathbf{x}_{L+U}\}$. Our goal is to find the best separating hyperplane characterized by $\theta = (\mathbf{w}, b)$, where $\mathbf{w}$ is the normal of the hyperplane and b is the bias term. We use separating hyperplanes to create binary codes and the sign of the following decision function defines the binary codes

$$f_\theta(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b. \tag{1}$$

The main idea of TSVM learning is to find an hyperplane that separates the labeled samples with a large margin at the same time ensures that the unlabeled samples will be as far as possible from the margin. So, both the labeled and unlabeled data play a dominant role for finding the separating hyperplane. To this end, earlier methods [2,14] used the following optimization formulation

$$\arg\min_{\mathbf{w},b} \ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{L} H_1(y_i(\mathbf{w}^\top \mathbf{x}_i + b)) + C^* \sum_{i=L+1}^{L+U} H_1(|\mathbf{w}^\top \mathbf{x}_i + b|), \tag{2}$$

where the function $H_1(t) = max(0, 1-t)$ is the classical Hinge loss plotted in Fig. 1, and $C(C^*)$ is a user defined parameter that controls the weight of errors associated to the labeled (unlabeled) data samples. The loss function for unlabeled data is shown in Fig. 2(a). It turned out the TSVM formulation given in (2) has the potential to assign all unlabeled samples to only one of the classes with a very large margin, which yields a poor classification accuracy. In order to solve this problem, a balancing constraint that enforces the unlabeled data to be assigned to both classes based on the same fraction of labeled data samples is introduced in [14]. Chapelle and Zien [2] used the following relaxed balancing constraint, which we also use in this study to create balanced binary hash codes

$$\frac{1}{U}\sum_{i=L+1}^{L+U} (\mathbf{w}^\top \mathbf{x}_i + b) \ = \ \frac{1}{L}\sum_{i=1}^{L} y_i. \tag{3}$$

Collobert et al. [3] replaced the symmetric Hinge loss of unalabeled points with the symmetric Ramp loss defined as

$$SR_s(t) \ = \ R_s(t) + R_s(-t), \tag{4}$$

where $R_s(t) = min(1-s, max(0, 1-t))$ is the Ramp Loss function illustrated in Fig. 1. Here $-1 < s \leq 0$ is a parameter that must be set by the user.

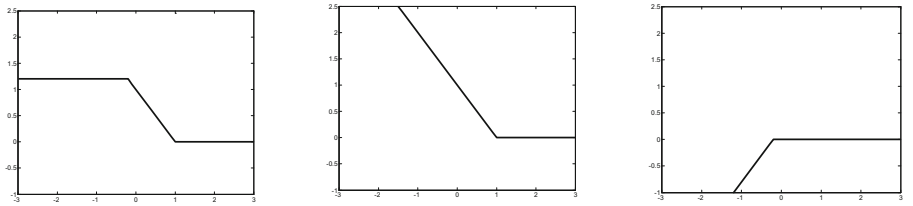


Fig. 1. The illustration of the Ramp loss function, $R_s(t) = H_1(t) - H_s(t)$, where $H_a(t) = max(0, a - t)$ is the classical Hinge loss. Here, we set $s = -0.20$.

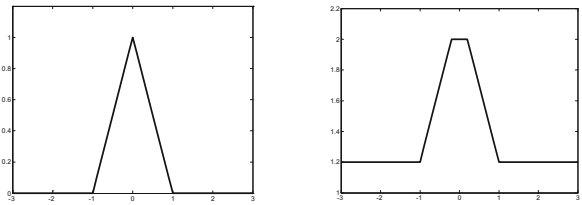
It should be noted that the loss functions for labeled and unlabeled data are not in the same range as shown in Figs. 1 and 2. For the symmetric Ramp loss used for unlabeled data, a sample can introduce at most a limited amount of cost value no matter of its position with respect to margin in the input space (the loss can be maximum 0.8 when s is set to -0.2). However, there is no bound for the Hinge loss used for labeled samples, e.g., a single outlying point farther from the margin can yield to a large loss. Therefore, the labeled outlying points – the samples that are misclassified outside the margin – start to play a dominant role in determining the separating hyperplane. As we mentioned earlier, labels can be very noisy in image retrieval applications, which aggravates the problem. To ameliorate this drawback, we interchange the convex Hinge loss with a more robust non-convex Ramp loss function. The Ramp loss also bounds the maximum amount of loss similar to the symmetric Ramp loss function and this helps to suppress the influence of misclassified examples. The superiority of the Ramp loss over the Hinge loss for supervised SVM training is well-proven and demonstrated in [4], so we adopt it to transductive learning here.

After these revisions, our robust TSVM method solves the following problem

$$\begin{aligned} \underset{\mathbf{w},b}{\arg\min}\ & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{L} R_s(y_i(\mathbf{w}^\top \mathbf{x}_i + b)) + C^* \sum_{i=L+1}^{L+U} SR_s(\mathbf{w}^\top \mathbf{x}_i + b) \\ \text{s.t. } & \frac{1}{U}\sum_{i=L+1}^{L+U}(\mathbf{w}^\top \mathbf{x}_i + b) = \frac{1}{L}\sum_{i=1}^{L} y_i. \end{aligned} \tag{5}$$

To use the symmetric Ramp loss function defined for unlabeled data samples, each unlabeled sample appears as two examples labeled with both negative and positive classes. More precisely, we create the new samples as follows

$$\begin{aligned} y_i &= +1, \quad i \in [L+1, \ldots, L+U], \\ y_i &= -1, \quad i \in [L+U+1, \ldots, L+2U], \\ \mathbf{x}_i &= \mathbf{x}_{i-U}, \quad i \in [L+U+1, \ldots, L+2U]. \end{aligned} \tag{6}$$



(a) (b)

Fig. 2. Loss functions used for unlabeled data: (a) $H_1(|t|) = max(0, 1 - |t|)$, (b) The symmetric Ramp loss, $SR_s(t) = R_s(t) + R_s(-t)$. Here, we set $s = -0.20$.

Then, by using the equations $R_s(t) = H_1(t) - H_s(t)$ and $SR_s(t) = R_s(t) + R_s(-t)$, the above cost function without constraint can be written as

$$J(\theta) = J_{convex}(\theta) + J_{concave}(\theta), \tag{7}$$

where

$$J_{convex}(\theta) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{L} H_1(y_i(\mathbf{w}^\top \mathbf{x}_i + b)) + C^* \sum_{i=L+1}^{L+2U} H_1(y_i(\mathbf{w}^\top \mathbf{x}_i + b)), \tag{8}$$

and

$$J_{concave}(\theta) = -C\sum_{i=1}^{L} H_s(y_i(\mathbf{w}^\top \mathbf{x}_i + b)) - C^* \sum_{i=L+1}^{L+2U} H_s(y_i(\mathbf{w}^\top \mathbf{x}_i + b)). \tag{9}$$

The above cost function (7) is not convex but it can be decomposed into a convex (8) and concave (9) part, so we can apply concave-convex procedure (CCCP) [33] to solve the problem. By employing CCCP, the minimization of $J(\theta)$ with respect to $\theta = (\mathbf{w}, b)$ can be achieved by iteratively updating the parameter θ by the following rule

$$\theta^{t+1} = \arg\min_{\theta}(J_{convex}(\theta) + J'_{concave}(\theta^t)\theta), \tag{10}$$

under the constraint $\frac{1}{U}\sum_{i=L+1}^{L+U}(\mathbf{w}^\top \mathbf{x}_i + b) = \frac{1}{L}\sum_{i=1}^{L} y_i$.

After some standard derivations given in Appendix (available at http://mlcv.ogu.edu.tr/pdf/appendix.pdf), the resulting final robust TSVM method can be summarized as in Algorithm 1. It should be noted that the optimization problem that constitutes the core of the CCCP is convex. Instead of taking dual of this convex problem and solving it with a dual QP solver as in [3], we consider the primal problem and use SG algorithm given in Algorithm 2 to solve it. Thus, the proposed method scales well with large-scale data. To initialize the method, we use supervised linear SVM trained with labeled data samples only.

Algorithm 1. The Robust Transductive Support Vector Machines (RTSVM)

Initialize $\theta^0 = (\mathbf{w}^0, b^0)$, $t = 0$, $\epsilon_1 > 0$, $\epsilon_2 > 0$
Compute

$$\beta_i^0 = y_i \frac{\partial J_{concave}(\theta)}{\partial f_\theta(\mathbf{x}_i)} = \begin{cases} C, & \text{if } y_i((\mathbf{w}^0)^\top \mathbf{x}_i + b^0) < s \text{ and } 1 \le i \le L \\ C^*, & \text{if } y_i((\mathbf{w}^0)^\top \mathbf{x}_i + b^0) < s \text{ and } L+1 \le i \le L+2U \\ 0, & \text{otherwise.} \end{cases}$$

while $||\mathbf{w}_{t+1} - \mathbf{w}_t|| \ge \epsilon_1$ or $||\boldsymbol{\beta}_{t+1} - \boldsymbol{\beta}_t|| \ge \epsilon_2$ **do**

– Solve the following convex minimization problem by using SG algorithm given in Algorithm 2

$\arg\min_{\mathbf{w},b} \ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{L} H_1(y_i(\mathbf{w}^\top \mathbf{x}_i + b)) + C^* \sum_{i=L+1}^{L+2U} H_1(y_i(\mathbf{w}^\top \mathbf{x}_i + b)) + \sum_{i=1}^{L+2U} \beta_i^t y_i(\mathbf{w}^\top \mathbf{x}_i + b)$
such that $\frac{1}{U}\sum_{i=1}^{L+U}(\mathbf{w}^\top \mathbf{x}_i + b) = \frac{1}{L}\sum_{i=1}^{L} y_i$;

– Set $\mathbf{w}^{t+1} = \mathbf{w}$, $b^{t+1} = b$;

– Compute

$$\beta_i^{t+1} = \begin{cases} C, & \text{if } y_i((\mathbf{w}^{t+1})^\top \mathbf{x}_i + b^{t+1}) < s \text{ and } 1 \le i \le L \\ C^*, & \text{if } y_i((\mathbf{w}^{t+1})^\top \mathbf{x}_i + b^{t+1}) < s \text{ and } L+1 \le i \le L+2U \\ 0, & \text{otherwise.} \end{cases}$$

– Set $t = t + 1$;

end while

2.2 Building Class Hierarchies

We use only labeled data to create class hierarchies. Assume that we are given some classes and corresponding labeled samples for each class (these are created by random selection of samples from each class or random selection of classes to create more independent hash functions). We use a binary hierarchical tree (BHT) that divides the image classes into two groups until each group consists of only one image class. In this setup, accuracy depends on the tree structure that creates well-balanced separable image class groups at each node of the tree. To this end, we use the Normalized Cuts (NCuts) algorithm of Shi and Malik [29] to split image classes into two groups (called positive and negative groups) since NCuts clustering algorithm maps the data samples into an infinite-dimensional feature space and cuts through the data by passing an hyperplane through the maximum gap in the mapped space [25]. In other words, it clusters the data into two balanced groups such that the margin between them is maximized. In our case, we must split image classes (not the individual image samples) into two groups. Therefore, we need to replace image data samples with image data classes. So, we approximate each class with a convex hull and use the convex

Algorithm 2. Stochastic Gradient Based Solver with Projection

Initialize
$\mathbf{w}_1$, b_1, $T > 0$, $\lambda_0 > 0$, $\epsilon > 0$
Description:
for $t \in 1, ..., T$ **do**
$\lambda_t \leftarrow \lambda_0/t$;
for $i \in$ randperm$(L + 2U)$ **do**
– Compute sub-gradients
$\mathbf{g}_t = \begin{cases} -y_i C(C^*)\mathbf{x}_i + \beta_i y_i \mathbf{x}_i, \text{ if } y_i(\mathbf{w}_t^\top \mathbf{x}_i + b_t) \leq 1 \\ \beta_i y_i \mathbf{x}_i, \qquad\qquad\qquad y_i(\mathbf{w}_t^\top \mathbf{x}_i + b_t) > 1. \end{cases}$
$h_t = \begin{cases} -y_i C(C^*) + \beta_i y_i, \text{ if } y_i(\mathbf{w}_t^\top \mathbf{x}_i + b_t) \leq 1 \\ \beta_i y_i, \qquad\qquad\quad y_i(\mathbf{w}_t^\top \mathbf{x}_i + b_t) > 1. \end{cases}$
– Update hyperplane parameters
$\tilde{\mathbf{w}}_t \leftarrow \mathbf{w}_t - \frac{\lambda_t}{L+2U}(\mathbf{w}_t + \mathbf{g}_t)$
$\tilde{b_t} \leftarrow b_t - \frac{\lambda_t}{L+2U} h_t$
– Project parameters onto the feasible set imposed by the constraint
$(\mathbf{w}_t, b_t) = \mathcal{P}(\tilde{\mathbf{w}}_t, \tilde{b_t})$
end for
if $(t > 2)$ & $(\|\mathbf{w}_t - \mathbf{w}_{t-1}\| < \epsilon)$, **break**
end for

hulls distances between image classes to create similarity matrix. It should be noted that convex hulls are largely used to approximate classes, e.g., the linear SVM uses convex hull modeling. In this setting, the edges, w_{ij}, of the similarity matrix $\mathbf{W}$ is computed as

$$w_{ij} = \begin{cases} \exp(-d(H_i^{convex}, H_j^{convex})/t), & \text{if } i \neq j \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

where t is the width of the Gaussian kernel function, and it must be set by the user. Note that the size of the similarity matrix is $C \times C$ where C is the number of classes. Thus, it is a much smaller sized matrix compared to other methods (mentioned at Introduction) using individual image samples. Then, we cluster the image classes into two groups by solving the generalized eigenvalue problem

$$\mathbf{L} = \lambda \mathbf{D}\mathbf{a}, \tag{12}$$

where $\mathbf{L} = \mathbf{D} - \mathbf{W}$ is the Laplacian matrix and $\mathbf{D}$ is a diagonal matrix whose entries are the column (or row) sums of $\mathbf{W}$. Finally, the components of the eigenvector $\mathbf{a}^*$ corresponding to the second smallest eigenvalue of (12) are thresholded to split image classes into two clusters, i.e.,

$$\begin{cases} y_i = -1, & \text{if } a_i^* \geq 0 \\ y_i = +1, & \text{if } a_i^* < 0 \end{cases} \tag{13}$$

Figure 3 illustrates the hierarchy obtained for 10-classes of CIFAR10 dataset. At the top node, it successfully separates man made vehicles (airplane, automobile,

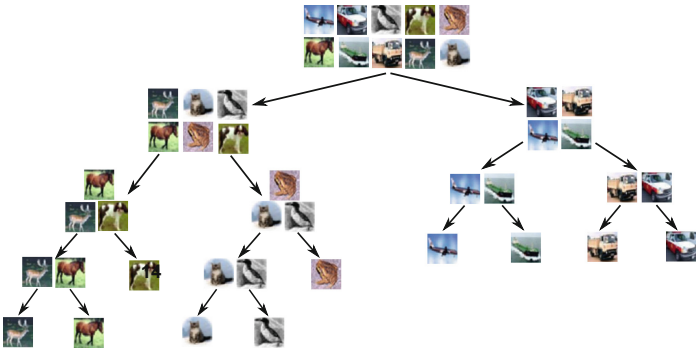


Fig. 3. Binary Hierarchical Tree obtained for CIFAR10 dataset using convex hull modeling of the classes. Each image represents an object class where it comes from.

ship, and truck) from the animals (bird, cat, deer, dog, frog, and horse). It also successfully groups visually similar groups such as automobile-truck, deer-horse, and airplane-ship together. So, our method produces both well-separated and well-balanced groups of classes, which is crucial for successful balanced binary hash codes. It should be noted this hierarchy is obtained automatically just by using the image feature samples and their labels. As mentioned earlier, more than one label can be assigned to image samples, e.g., assume that an image sample contains both *people* and *car*. In such cases, we treat groups with multiple labels as a new category and manually set the similarities between the related classes (*people* and *car* classes) to maximum. By doing so, we postpone to separate these related classes by grouping them as similar classes. So, they appear at the lower nodes of the class hierarchy where we can do a finer separation between them.

2.3 Creating Binary Hash Codes

Once we split image classes into two groups at each node of the BHT, we run TSVM algorithm by using both labeled and randomly chosen unlabeled data to find the separating hyperplanes. Then these hyperplanes can be used in two ways to produce hash codes. In the first place, we can use the following rule to create hash codes

$$h_i(\mathbf{x}) = \begin{cases} 1 & \text{when } \mathbf{w}_i^\top \mathbf{x} + b_i \geq 0 \\ 0 & \text{when } \mathbf{w}_i^\top \mathbf{x} + b_i < 0 \end{cases} \tag{14}$$

where $\mathbf{w}_i$s are the returned hyperplane normals and b_is are the corresponding bias parameters. Each BHT produces $C-1$ hash functions where C is the number of classes used to build BHT. So, the total number of hash bits will be $C-1$ times the number of BHTs. As a second choice, we can use hyperplane normals to embed the data samples onto a more discriminative space and then use an

Euclidean distance preserving hashing method (e.g., LSH) in the embedded space (this can be seen as metric learning followed by using a hashing method that approximates the learned distance metric). Lastly, we use Hamming distance to find the distances between hash codes, but weighted Hamming distances using hierarchy or margin can be also be used for this goal.

3 Experiments

Here, we conduct image retrieval experiments on three datasets. We compared the proposed hashing method, TSVMH-BHT (Transductive Support Vector Machine Hashing using Binary Hierarchical Tree), with both the supervised and unsupervised hashing methods: LSH [6], PCA-RR (Principal Component Analysis – Random Rotations) [7], PCA-ITQ (Principal Component Analysis – Iterative Quantization) [7], SKLSH (Shift-Invariant Kernel Locality Sensitive Hashing) [26], SH (Spectral Hashing) [31], SHD (Spherical Hamming Distance) [10], and SDH (Supervised Hashing) [21]. In addition to these hashing methods we also report the results obtained using PQ (Product Quantization) method of Jegou et al. [12] as a baseline. We also give the best reported accuracies of recent hashing methods using deep neural networks.

3.1 Experiments on CIFAR10 Dataset

Cifar10 dataset (available at http://www.cs.toronto.edu/~kriz/cifar.html) includes 60 K 32×32 small images of 10 objects: airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. 50 K samples are used as training and they are split into 5 batches whereas the remaining 10 K samples are used for testing. We first used gray-scale GIST descriptors computed at three different scales (8,8,4), resulting in 320-dimensional image feature vectors as in [7]. But, the nearest-neighbor accuracy of this primitive feature representation was too small so we also used 16384 dimensional fisher vectors (FVs) and 4096 dimensional CNN features which significantly outperformed GIST descriptors in our experiments. We used a similar setup as in [28] to extract FVs. More precisely, we extracted many descriptors per image from 12×12 patches on a regular grid every two pixels at 3 scales. The dimensionality of the tested descriptors is reduced to 80 by using Principal Component Analysis (PCA), and 128-component Gaussian mixture model (GMM) components are used to obtain FVs. To extract CNN features, all images are first resized to 256×256 and then we used Caffe [13] implementation of the CNN described by Krizhevsky et al. [16] by using the identical setting used for ILSVRC 2012 classification with the exception that the base learning rate was set to 0.001. We used 80 % of the full training data for training and the remaining 20 % as validation to train the CNN classifier. The number of iterations is set to 120 K.

For all methods, we used the full training data to create hash functions, but we use only the samples in each batch to find the Hamming distances from the test samples. So, the results are averages over the results of 5 trials obtained

Table 1. mAP Scores (%) for CIFAR 10 dataset using GIST and FVs

GIST	32 bits	64 bits	128 bits	256 bits
TSVMH-BHT	**37.20**	**39.67**	**41.27**	**41.78**
SDH	34.51	36.64	37.88	38.59
LSH	23.10	24.57	26.25	26.38
SH	19.45	19.69	19.54	19.19
SHD	21.81	24.02	25.82	27.14
SKLSH	15.23	17.29	19.43	21.26
PCA-ITQ	24.51	26.08	27.10	28.05
PCA-RR	16.70	18.77	20.08	21.73
PQ	27.10	27.60	28.10	28.50
FVs	32 bits	64 bits	128 bits	256 bits
TSVMH-BHT	**46.74**	**51.37**	**53.94**	**55.10**
SDH	31.66	33.52	34.62	35.36
LSH	19.43	20.57	20.53	21.57
SH	19.60	20.43	21.44	21.86
SHD	19.47	21.91	24.09	25.77
SKLSH	11.10	11.38	11.93	12.68
PCA-ITQ	24.59	26.03	27.34	28.05
PCA-RR	23.07	24.53	25.83	36.76
PQ	24.30	23.80	22.90	22.00

Table 2. mAP Scores (%) for CIFAR 10 dataset using CNN features

CNN	32 bits	64 bits	128 bits	256 bits
TSVMH-BHT	79.97	81.89	82.45	82.79
SDH	**83.05**	**83.59**	**83.77**	**83.83**
LSH	73.98	74.87	75.56	76.40
SH	65.74	67.15	65.04	60.52
SHD	65.41	66.38	64.83	64.21
SKLSH	40.79	56.33	64.17	67.85
PCA-ITQ	80.78	81.27	81.86	82.05
PCA-RR	74.19	77.15	78.46	79.15
PQ	79.88	80.30	80.40	80.65
[19]	55.80	– –	– –	– –
[32]	52.10	– –	– –	– –
[35]	62.53	62.81	– –	– –
[36]	**≈86.0**	**≈86.0**	**≈86.0**	– –

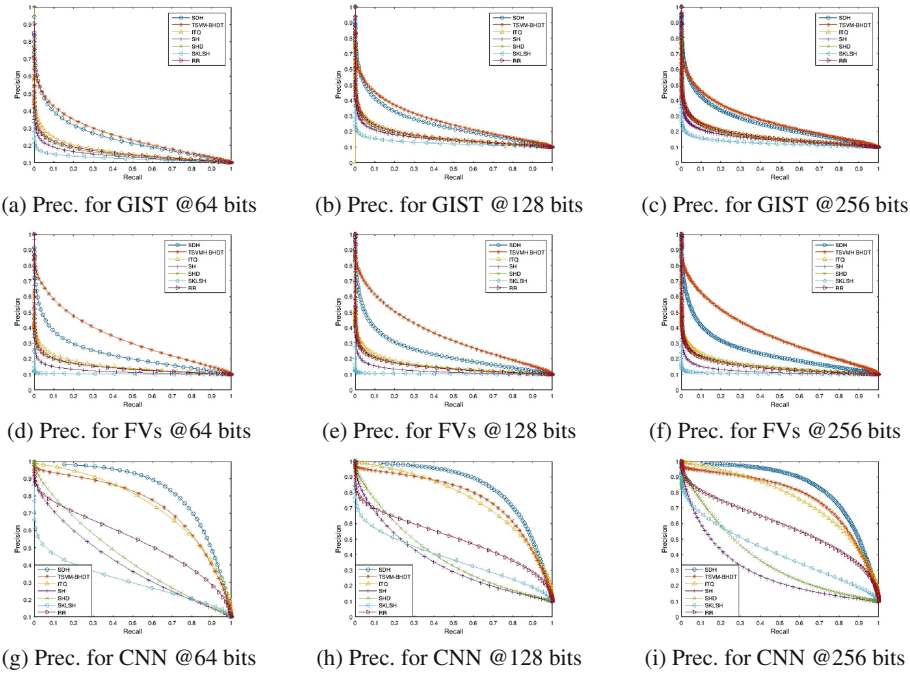
for each training batch. We used randomly chosen 600 labeled samples and 900 unlabeled samples from each class to train our method. It should be noted the total training set size is 15 K and supervised hashing methods and unsupervised hashing methods that build dense similarity (or kernel) matrix cannot be used directly even for this moderate sized dataset. Therefore, randomly chosen anchor points are used in SDH [21]. The default value for the number of anchor points is 300 for SDH, but we increased it to 500 for better accuracies.

The mAP (mean Average Precision) scores using class labels as ground truth are given in Table 1 and Fig. 4 illustrates Precision curves obtained for different bit sizes. The mAP scores are obtained by using top 500 returned images as a function of code size as in [7]. [7] also reports mAP scores using the Euclidean distances as ground truth, but this is wrong in our opinion since the performance of the Euclidean distance is very poor: Euclidean Distance in the original input space yields to 27.15 % mAP for GIST, 28.22 % for FVs and 76.90 % for CNNs. As can be seen in Table 1, both supervised methods SDH and TSVMH-BHT dominate other unsupervised methods for GIST and FVs and our proposed method TSVMH-BHT achieves the best accuracies. The difference between the accuracies of these two methods is small for GIST but our proposed method significantly outperforms SDH for FVs. The mAP score for the proposed method is approximately 20 % better than the second best method SDH when 256 bits are used, which undoubtedly shows that the proposed method is better suited for high-dimensional visual image representations.

Table 2 shows the accuracies obtained using CNN features and some recently reported accuracies obtained using deep neural networks in the literature. It should be noted that the CNN features we extracted are different than the ones obtained by the methods given under PQ method in Table 2 since these methods are trained end-to-end fashion. Yet our proposed method outperforms all of them except the one given in [36]. SDH slightly beats our proposed method. The unsupervised PCA-ITQ also works well. [20] reports 89.4 % mAP accuracy for Cifar10 dataset. We verified this result by using their pre-trained models. But, their training files show that they used test data as validation data to train the CNN network, which is a violation of a fair testing procedure. Thus, we omitted this result in our table.

3.2 Experiments on NUS-WIDE Dataset

The NUS-Wide dataset has approximately 270 K images collected from Flickr. Each image is annotated with one or multi labels in 81 semantic classes. To comapre our results to the literature, we follow the settings in [19,22,32,36] and we use the images associated with the 21 most frequent labels. The number of final training images is 96638 and the number of test images is 64704. Two images are considered as a true match if they share at least one common label as in [19, 22,32,36]. We used both FVs and CNN feature vectors for representing images. To obtain CNN feaures we first resized images to 256×256 as before and used the same setting we used in Cifar10. Note that this is a multi-label dataset, thus we selected images with non-overlapping unique labels to train the classifier network.



(a) Prec. for GIST @64 bits (b) Prec. for GIST @128 bits (c) Prec. for GIST @256 bits

(d) Prec. for FVs @64 bits (e) Prec. for FVs @128 bits (f) Prec. for FVs @256 bits

(g) Prec. for CNN @64 bits (h) Prec. for CNN @128 bits (i) Prec. for CNN @256 bits

Fig. 4. Comparisons of the hashing methods on CIFAR10 dataset using labels as ground truth.

But, results were very low compared to the ones obtained for FVs. We believe that the results were low mainly because the loss function is not designed for multi-label data. The noisy labels was another reason for the low accuracy. Thus, we used pre-trained Caffe model of ILSVRC 2012 to extract 4096-dimensional CNN features. Results are given in Table 3. Using Euclidean distance with full features yields to 52.20 % mAP for FVs and 69.65 % for CNN features. As in the previous case, hashing codes obtained CNN features yield to better accuracies. The proposed method achieves the best accuracies in all cases except for 32 bits and improves the Euclidean distance metric for 64 bits and above. The reported results for the best performing method [36] on Cifar10 are very low for NUS-WIDE. To the best our knowledge, our results are the best published mAP scores for NUS-WIDE dataset. As before, the performance difference is very significant for FVs. We believe that our better accuracies compared to the other state-of-art methods are due to the using robust Ramp loss for noisy labels.

3.3 Experiments on MNIST Dataset

The MNIST[1] digit dataset consists of 70 K hand-written digit samples, each of size 28×28 pixels. For this dataset, 60 K samples are allocated for training and

[1] Available at http://yann.lecun.com/exdb/mnist/.

Table 3. mAP Scores (%) for NUS-WIDE dataset using FVs and CNN features

FVs	32 bits	64 bits	128 bits	256 bits
TSVMH-BHT	53.97	**57.73**	**61.76**	**63.99**
SDH	**54.20**	56.41	58.14	58.93
LSH	30.25	36.27	39.50	43.20
SH	46.31	45.97	47.60	48.51
SHD	47.08	50.09	52.41	52.89
SKLSH	33.24	35.25	36.50	37.68
PCA-ITQ	50.41	51.95	52.63	53.17
PCA-RR	49.89	51.17	52.09	52.95
PQ	30.75	31.74	33.49	40.19
CNN	32 bits	64 bits	128 bits	256 bits
TSVMH-BHT	68.58	**72.90**	**74.64**	**76.21**
SDH	69.09	72.41	73.71	74.75
LSH	66.78	68.91	69.47	69.50
H	58.03	60.08	61.31	64.41
SHD	61.60	64.23	64.83	64.66
SKLSH	39.65	43.37	47.47	53.65
PCA-ITQ	68.62	70.91	72.48	73.55
PCA-RR	65.82	68.27	70.57	71.60
PQ	**71.07**	71.93	72.20	72.17
[19]	71.30	– –	– –	– –
[32]	62.90	– –	– –	– –
[35]	62.64	63.82	– –	– –
[36]	≈52.0	≈52.5	≈54.0	– –

Table 4. mAP Scores (%) for MNIST Digit dataset

Methods	32 bits	64 bits	128 bits	256 bits
TSVMH-BHT	**87.68**	**89.15**	**89.07**	**89.13**
SDH	81.29	85.37	86.10	86.26
LSH	72.15	74.40	79.30	82.43
SH	70.44	72.28	73.87	74.15
SHD	67.86	74.06	75.98	76.58
SKLSH	31.36	47.12	63.04	74.08
PCA-ITQ	79.55	83.37	85.50	86.23
PCA-RR	67.91	75.38	79.16	83.05
PQ	85.40	85.40	85.20	85.40

the remaining 10 K samples are reserved for test. We use gray-scale values as visual features, thus dimensionality of the sample space is 784. We use randomly chosen 5000 labeled samples and 800 unlabeled samples from each class to train the proposed method. It should be noted that test samples are not used as unlabeled samples during training. Results are given in Table 4. Euclidean Distance in the original input space yields to 85.95 % mAP score which is quite satisfactory. Yet, our proposed method gives better accuracies than NN for all bit sizes. SDH is the second best performing method and it can improve NN accuracy only for 128 bits and above.

4 Conclusion

In this study, we discussed the fact that the Euclidean distances in the high-dimensional feature spaces can be misleading, thus hashing methods approximating the Euclidean distances may perform poorly. To counter this, we proposed a hashing method that does both metric learning and fast image search. To this end, we used binary hierarchical trees and TSVM classifier. We proposed a more robust TSVM method designed for especially image retrieval applications. Using TSVM is extremely important here since it also exploits the unlabeled data that is neglected by many related hashing and distance metric learning methods.

We tested the proposed method on three image retrieval datasets. The results with high-dimensional FV features were particularly promising: Our method significantly outperformed all other tested hashing methods with FVs. We also obtained state-of-art results using CNN features on noisy labeled NUS-WIDE dataset which shows the importance of using our robust Ramp loss function. We also compared the proposed method to the recently published hashing methods using deep neural networks. These methods emphasize the importance of simultaneous learning of image features and binary codes, yet the results prove that this issue has not been resolved yet since majority of these deep neural networks methods yield very low accuracies compared to our method which learns hash codes independently from the pre-computed CNN features.

Acknowledgment. This work has been supported by the Scientific and Technological Research Council of Turkey (TUBİTAK) under Grant number TUBITAK-113E118.

References

1. Cevikalp, H., Triggs, B., Polikar, R.: Nearest hyperdisk methods for high-dimensional classification. In: ICML (2008)
2. Chapelle, O., Zien, A.: Semi-supervised classification by low density separation. In: Proceedings of 10th International Workshop on Artifical Intelligence and Statistics (2005)
3. Collobert, R., Sinz, F., Weston, J., Bottou, L.: Large scale transductive SVMs. J. Mach. Learn. Res. **7**, 1687–1712 (2006)
4. Ertekin, S., Bottou, L., Giles, C.L.: Nonconvex online support vector machines. IEEE Trans. Pattern Anal. Mach. Intell. **33**, 368–381 (2011)

5. Fergus, R., Bernal, H., Weiss, Y., Torralba, A.: Semantic label sharing for learning with many categories. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010. LNCS, vol. 6311, pp. 762–775. Springer, Heidelberg (2010). doi:10.1007/978-3-642-15549-9_55
6. Gionis, A., Indyk, P., Motwani, R.: Similarity search in high dimensions via hashing. In: International Conference on Very Large Databases (1999)
7. Gong, Y., Lazebnik, S., Gordo, A., Perronnin, F.: Iterative quantization: a procrustean approach to learning binary codes for large-scale image retrieval. IEEE Trans. PAMI **35**, 2916–2929 (2013)
8. Gong, Y., Jia, Y., Leung, T., Toshev, A., Ioffe, S.: Deep convolutional ranking for multilabel image annotation. arXiv:1312.4894 (2013)
9. He, X., Cai, D., Han, J.: Learning a maximum margin subspace for image retrieval. IEEE Trans. Knowl. Data Eng. **20**, 189–201 (2008)
10. Heo, J.P., Lee, Y., He, J., Chang, S.F., Yoon, S.E.: Spherical hashing. In: CVPR (2012)
11. Hoi, S.C.H., Jin, R., Zhu, J., Lyu, M.R.: Semi-supervised svm batch mode active learning for image retrieval. In: CVPR (2008)
12. Jegou, H., Douze, M., Schmid, C.: Product quantization for nearest neighbor search. IEEE Trans. PAMI **33**, 117–128 (2010)
13. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093 (2014)
14. Joachims, T.: Transductive inference for text classification using support vector machines. In: International Conference on Machine Learning (1999)
15. Joly, A., Buisson, O.: Random maximum margin hashing. In: CVPR (2011)
16. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS (2012)
17. Kulis, B., Darrell, T.: Learning to hash with binary reconstructive embeddings. In: NIPS (2009)
18. Kulis, B., Grauman, K.: Kernelized locality-sensitive hashing for scalable image search. In: ICCV (2009)
19. Lai, H., Pan, Y., Yan, S.: Simultaneous feature learning and hash coding with deep neural networks. In: CVPR (2015)
20. Lin, K., Yang, H.F., Hsiao, J.H., Chen, C.S.: Deep learning of binary hash codes for fast image retrieval. In: IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) (2015)
21. Liu, W., Wang, J., Ji, R., Jiang, Y.G., Chang, S.F.: Supervised hashing with kernels. In: CVPR (2012)
22. Liu, W., Wang, J., Kumar, S., Chang, S.F.: Hashing with graphs. In: ICML (2011)
23. Mu, Y., Shen, J., Yan, S.: Weakly-supervised hashing in kernel space. In: CVPR (2010)
24. Norouzi, M., Fleet, D.J.: Minimal loss hashing for compact binary codes. In: ICML (2011)
25. Rahimi, A., Recht, B.: Clustering with normalized cuts is clustering with a hyperplane. In: Statistical Learning in Computer Vision (2004)
26. Raginsky, M., Lazebnik, S.: Locality-sensitive binary codes from shift-invariant kernels. In: ICCV (2009)
27. Salakhutdinov, R., Hinton, G.: Semantic hashing. Int. J. Approximate Reason. **50**(12), 969–978 (2009)

28. Sanchez, J., Perronnin, F., Mensink, T., Verbeek, J.: Image classification with the fisher vector: theory and practice. Int. J. Comput. Vis. **34**, 1704–1716 (2013)
29. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Trans. PAMI **22**, 888–905 (2000)
30. Wang, J., Kumar, S., Chang, S.F.: Semi-supervised hashing for scalable image retrieval. In: CVPR (2010)
31. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing. In: NIPS (2008)
32. Xia, R., Pan, Y., Lai, H., Liu, C., Yan, S.: Supervised hashing for image retrieval via image representation learning. In: Proceedings of the 28 AAAI Conference on Artificial Intelligence (2014)
33. Yullie, A.L., Rangarajan, A.: The concave-convex procedure (cccp). In: Neural Information Processing Systems (2002)
34. Zhang, L., Wang, L., Lin, W.: Semi-supervised biased maximum margin analysis for interactive image retrieval. IEEE Trans. Image Process. **21**, 2294–2308 (2012)
35. Zhang, R., Lin, L., Zhang, R., Zuo, W., Zhang, L.: Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification. IEEE Trans. Image Process. **24**, 4766–4779 (2015)
36. Zhang, Z., Chen, Y., Saligrama, V.: Supervised hashing with deep neural networks. arXiv:1511.04524 (2015)
37. Zhao, F., Huang, Y., Wang, L., Tan, T.: Deep semantic ranking based hashing for multi-label image retrieval. In: CVPR (2015)