Instance-Aware Hashing for Multi-Label Image Retrieval

Hanjiang Lai, Pan Yan, Xiangbo Shu, Yunchao Wei, and Shuicheng Yan, *Senior Member, IEEE*

***Abstract*— Similarity-preserving hashing is a commonly used method for nearest neighbor search in large-scale image retrieval. For image retrieval, deep-network-based hashing methods are appealing, since they can simultaneously learn effective image representations and compact hash codes. This paper focuses on deep-network-based hashing for multi-label images, each of which may contain objects of multiple categories. In most existing hashing methods, each image is represented by one piece of hash code, which is referred to as semantic hashing. This setting may be suboptimal for multi-label image retrieval. To solve this problem, we propose a deep architecture that learns instance-aware image representations for multi-label image data, which are organized in multiple groups, with each group containing the features for one category. The instance-aware representations not only bring advantages to semantic hashing but also can be used in category-aware hashing, in which an image is represented by multiple pieces of hash codes and each piece of code corresponds to a category. Extensive evaluations conducted on several benchmark data sets demonstrate that for both the semantic hashing and the category-aware hashing, the proposed method shows substantial improvement over the state-of-the-art supervised and unsupervised hashing methods.**

***Index Terms*— Multi-label, image retrieval, instance-aware image representation, category-aware hashing, semantic hashing, deep learning.**

I. Introduction

LARGE-SCALE image retrieval, which is to find images containing similar objects as in a query image, has attracted increasing interest due to the ever-growing amount of available image data on the Web. Similarity-preserving hashing is a popular nearest neighbor search technique for image retrieval on datasets with millions or even billions of images.

Manuscript received July 16, 2015; revised November 16, 2015 and January 9, 2016; accepted March 8, 2016. Date of publication March 22, 2016; date of current version April 14, 2016. This work was supported in part by the Natural Science Foundation of Guangdong Province under Grant S2013010011905 and in part by the National Natural Science Foundation of China under Grant 61370021, Grant U1401256, Grant 61472453, and Grant U1501252. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Dacheng Tao. *(Corresponding author: Yan Pan.)*

H. Lai and P. Yan are with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China (e-mail: laihanj@gmail.com; panyan5@mail.sysu.edu.cn).

X. Shu is with the School of Computer Science and Technology, Nanjing 210094, China (e-mail: shuxb104@gmail.com).

Y. Wei is with the Institute of Information Science, Beijing Jiaotong University, Beijing 100044, China (e-mail: wychao1987@gmail.com).

S. Yan is with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 119077 (e-mail: eleyans@nus.edu.sg).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2016.2545300

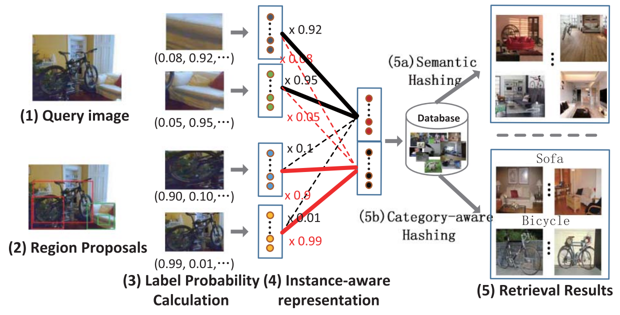


Fig. 1. Illustration of instance-aware image retrieval. (1) Given a query image, e.g., containing a bicycle and a sofa, the proposal method (2) generates region proposals, (3) computes the label probability scores for each proposal, (4) encodes each proposal to an intermediate feature vector, and then computes the weighted average of these vectors (with the label probability scores being the weights) to generate the instance-aware representations organized in multiple groups, each corresponding to an object. After that, this representation is converted to (5a) one piece of hash code for semantic hashing or (5b) multiple pieces of hash codes, each piece corresponding to a category, for category-aware hashing.

A representative stream of similarity-preserving hashing is learning-to-hash, i.e., learning to compress data points (e.g., images) into binary representations such that semantically similar data points have nearby binary codes. The existing learning-to-hash methods can be divided into two main categories: unsupervised methods and supervised methods. Unsupervised methods (e.g., [1]–[3]) learn a set of hash functions from unlabeled data without any side information. Supervised methods (e.g., [4]–[7]) try to learn compact hash codes by leveraging supervised information on data points (e.g., similarities on pairs of images). Among various supervised learning-to-hash methods for image retrieval, an emerging stream is deep-networks-based hashing that learns bitwise codes as well as image representations via carefully designed deep neural networks. Several deep-networks-based hashing methods have been proposed (e.g., [4], [8], [9]).

Multi-label images, each of which may contain objects of multiple categories, are widely involved in many image retrieval systems. However, in most existing hashing methods for images, the semantic similarities are defined at image level, and each image is represented by one piece of hash code. This setting may be suboptimal for multi-label image retrieval.

In this paper, we consider **instance-aware retrieval** for multi-label image data, which includes semantic hashing [10] and category-aware hashing. Specifically, given a multi-label query image, a natural demand is to organize the retrieved results in groups, each group corresponding to one category. For example, as shown in Figure 1, given a query

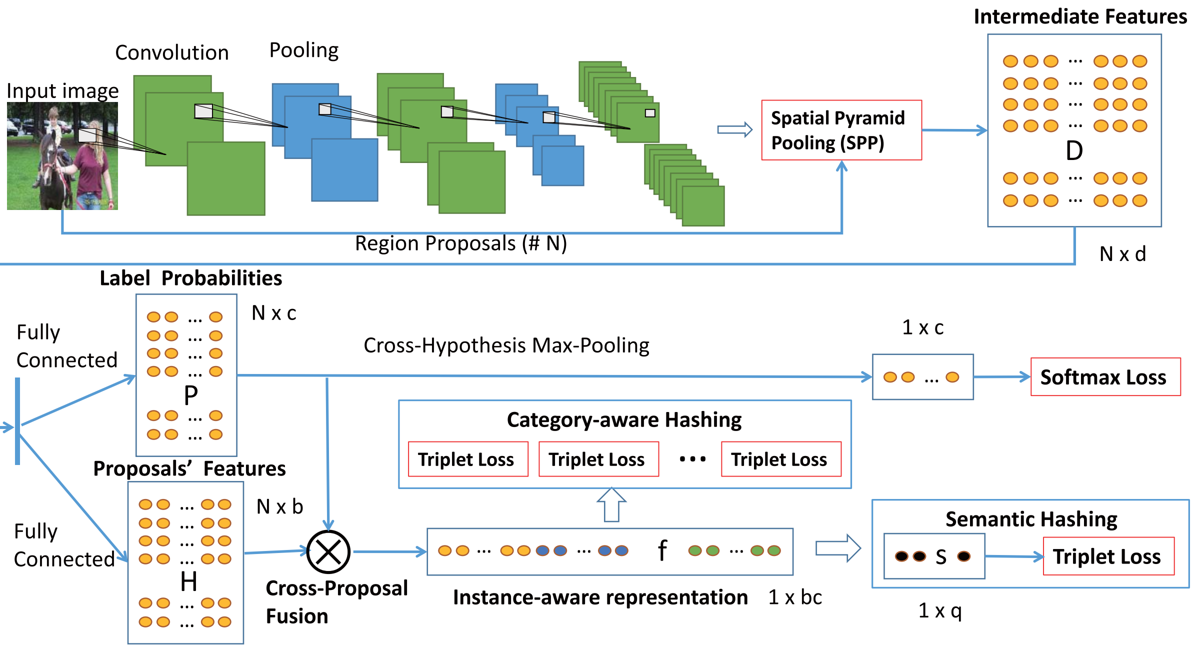


Fig. 2. Overview of the proposed deep architecture for hashing on multi-label images. The proposed architecture takes an image (e.g, in c classes) and its automatically generated N region proposals as inputs. The image firstly goes trough the deep convolution sub-network, and then N intermediate feature vectors are generated for the region proposals via the spatial pyramid pooling scheme. With the intermediate features, our network is divided into two branches: one for calculating the label probabilities of the region proposals (see Fig. 3), and the other for generating the proposals' features. Cross-proposal fusion is performed to merge the proposals' features (with label probabilities) into an intermediate multiple-slice representation (i.e., f) in which each slice corresponds to one category (see Fig. 4). After that, this intermediate representation is converted to multiple pieces of hash codes (for category-aware hashing) or one piece of hash code (for semantic hashing).

image containing a *bicycle* and a *sofa*, one would like to organize the retrieved results in two groups: each image in the first (second) group contains a bicycle (sofa) similar to the one in the query image. In order to achieve instance-aware retrieval, we propose a new image representation organized in groups, by incorporating automatically generated candidate object proposals and label probability calculation into the learning process. Figure 1 shows an example for the generation of the instance-aware image representation.

More specifically, we propose a deep neural network that simultaneously learns binary hash codes and the representation tailored for multi-label image data. As shown in Figure 2, the proposed architecture has four building blocks: 1) a set of N automatically generated candidate object proposals in the form of bounding boxes, as inputs to the deep neural network; 2) stacked convolutional layers to capture the features of the input proposals, followed by a Spatial Pyramid Pooling (SPP) layer [11] to map each proposal to a d-dimensional intermediate representation; 3) a label probability calculation module that maps the intermediate representation to the image labels (in c classes), which leads to an $N \times c$ probability matrix with the i-th row representing the label probabilities of the i-th proposal belonging to each class; 4) a hash coding module, where firstly an instance-aware representation is captured, in which the probability matrix in the third module is used as the input, and then either category-aware hash codes or semantic hash codes are generated based on this representation.

The proposed deep architecture can be used to generate hash codes for category-aware hashing, where an image is represented by multiple pieces of hash codes, each of which corresponds to a category. In addition, we show that the proposed image representation can improve the quality of semantic hashing in which an image is represented by one piece of hash code.

Our contributions in this paper can be summarized as follows. First, we propose a deep architecture that can generate hash codes for instance-aware retrieval. To the best of our knowledge, we are the first to conduct instance-aware retrieval via learning-based hashing. Second, we propose to incorporate automatically generated candidate object proposals and label probability calculation in the proposed deep architecture. We empirically show that the proposed method has superior performance gains over several state-of-the-art hashing methods.

II. Related Work

Due to the encouraging search speed, hashing has become a popular method for nearest neighbor search in large-scale image retrieval.

Hashing methods can be divided into data independent hashing and data dependent hashing. The early efforts mainly focus on data independent hashing. For example, the notable Locality-Sensitive Hashing (LSH) [12] method constructs hash functions by random projections or random permutations that are independent of the data points. The main limitation of data independent methods is that they usually require long hash codes to obtain good performance. However, long hash codes lead to inefficient search due to the required large storage space and the low recall rates.

Learning-based hashing (or Learning-to-hash) pursues a compact binary representation from the training data. Based on whether side information is used or not, learning-to-hash methods can be divided into two categories: unsupervised methods and supervised methods.

Unsupervised methods try to learn a set of similarity-preserving hash functions only from the unlabeled data. Representative methods in this category include Kernelized LSH (KLSH) [2], Semantic hashing [13], Spectral hashing [14], Anchor Graph Hashing [3], and Iterative Quantization (ITQ) [1]. Kernelized LSH (KLSH) [2] generalizes LSH to accommodate arbitrary kernel functions, making it possible to learn hash functions which preserve data points' similarity in a kernel space. Semantic hashing [13] generates hash functions by a deep auto-encoder via stacking multiple restricted Boltzmann machines (RBMs). Graph-based hashing methods, such as Spectral hashing [14] and Anchor Graph Hashing [3], learn non-linear mappings as hash functions which try to preserve the similarities within the data neighborhood graph. In order to reduce the quantization errors, Iterative Quantization (ITQ) [1] seeks to learn an orthogonal rotation matrix which is applied to the data matrix after principal component analysis projections.

Supervised methods aim to learn better bitwise representations by incorporating supervised information. Notable methods in this category include Binary Reconstruction Embedding (BRE) [6], Minimal Loss Hashing (MLH) [15], Supervised Hashing with Kernels (KSH) [5], Column Generation Hash (CGHash) [16], and Semi-Supervised Hashing (SSH) [17]. Binary Reconstruction Embedding (BRE) [6] learns hash functions by explicitly minimizing the reconstruction errors between the original distances of data points and the Hamming distances of the corresponding binary codes. Minimal Loss Hashing (MLH) [15] learns similarity-preserving hash codes by minimizing a hinge-like loss function which is formulated as structured prediction with latent variables. Supervised Hashing with Kernels (KSH) [5] is a kernel-based supervised method which learns to hash the data points to compact binary codes whose Hamming distances are minimized on similar pairs and maximized on dissimilar pairs. Column Generation Hash (CGHash) [16] is a column generation based method to learn hash functions with proximity comparison information. Semi-Supervised Hashing (SSH) [17] learns hash functions via minimizing similarity errors on the labeled data while simultaneously maximizing the entropy of the learnt hash codes over the unlabeled data. In most image retrieval applications, the number of labeled positive samples is small, which results in bias towards the negative samples and over-fitting. Tao *et al.* [18] proposed an asymmetric bagging and random subspace SVM (ABRS-SVM) to handle these problems.

In supervised hashing methods for image retrieval, an emerging stream is the deep-networks-based methods [4], [8], [9], [19] which learn image representations as well as binary hash codes. Xia *et al.* [4] proposed Convolutional-Neural-Networks-based Hashing (CNNH), which is a two-stage method. In its first stage, approximate hash codes are learned from the supervised information. Then, in the second stage, hash functions are learned based on those approximate hash codes via deep convolutional networks. Lai *et al.* [8] proposed a one-stage hashing method that generates bitwise hash codes via a carefully designed deep architecture. Zhao *et al.* [9] proposed a ranking based hashing method for learning hash functions that preserve multi-level semantic similarity between images, via deep convolutional networks. Lin *et al.* [20] proposed to learn the hash codes and image representations in a point-wised manner, which is suitable for large-scale datasets. Wang *et al.* [21] proposed Deep Multimodal Hashing with Orthogonal Regularization (DMHOR) method for multimodal data. All of these methods generate one piece of hash code for each image, which may be inappropriate for multi-label image retrieval. Different from the existing methods, the proposed method can generate multiple pieces of hash codes for an image, each piece corresponding to a(n) instance/category.

III. The Proposed Method

Our method consists of four modules. The first module is to generate region proposals for an input image. The second module is to capture the features for the generated region proposals. It contains a deep convolution sub-network followed by a Spatial Pyramid Pooling layer [11]. The third module is a label probability calculation module, which outputs a probability matrix whose i-th row represents the probability scores of the i-th proposal belonging to each class. The fourth module is a hash coding module that firstly generates the instance-aware representation, and then converts this representation to hash codes for either category-aware hashing or semantic hashing. In the following, we will present the details of these modules, respectively.

A. Region Proposal Generation Module

Many methods for generating category-independent region proposals have been proposed, e.g., Constrained Parametric Min-Cuts (CPMC) [22], Selective Search [23], Multi-scale Combinatorial Grouping (MCG) [24], BInarized Normed Gradients (BING) [25] and Geodesic Object Proposals (GOP) [26]. In this paper, we use GOP [26] to automatically generate region proposals for an input. Note that other methods for region proposal generation can also be used in our framework.

GOP is a method that can generate both segmentation masks and bounding box proposals. We use the code[1] provided by the authors to generate the bounding boxes for region proposals.

B. Deep Convolution Sub-Network Module

GoogLeNet [27] is a recently proposed deep architecture that has shown its success in object categorization and object detection. The core of GoogLeNet is the Inception-style convolution module which allows increasing the depth and width of the network while keeping reasonable computational costs. Here we adopt the architecture of GoogLeNet as our basic framework to compute the features for the input proposals.

[1] http://www.philkr.net/home/gop

Since the GoogLeNet is a very deep network and has many layers, we use the pre-trained GoogLeNet model[2] to initialize its weights which can be regarded as regularization [28] and help its generalization.

However, since the number of generated region proposals for an input image may be large (e.g., more than 1000), it is computationally expensive if one directly uses GoogLeNet to extract features from these proposals. This is unaffordable for hashing-based retrieval since the retrieval system may need a long time to respond to a query.

To address this issue, we use the "Spatial Pyramid Pooling" (SPP) scheme [11]. The advantage of using SPP is that we can compute the feature map from the entire input image only once. Then, with this feature map, we pool features in each generated region proposal to generate a fixed-length representation. Using SPP, we avoid repeatedly computing features for the input region proposals via a deep convolutional network. Specifically, as shown in Figure 2, we add an SPP layer after the last convolutional layer of GoogLeNet. We assume that each input image has N automatically generated region proposals. For each input region proposal, we encode its top-left and bottom-right coordinates to a 4D vector $\mathbf{L}_i \in \mathbb{R}^4 (i = 1, 2, \ldots, N)$. The elements in this vector are scaled to [0, 1] by dividing the width/height of the image, making them invariant to the absolute image size. With this 4-dimensional vector as the input, the SPP layer generates a fixed-length feature vector for the corresponding proposal. Through the SPP layer, we assume that each proposal is mapped to a d-dimensional intermediate feature vector. Hence, for each input image, the output of this module is an $N \times d$ matrix.

After this module, the network is divided into two branches: one for the label probability calculation module, and the other for the hash coding module.

C. Label Probability Calculation Module

In this subsection, we will show how to learn the label probability for each region proposal. Suppose there are c class labels, and a probability vector is generated for each proposal, e.g., $P^i = (P^i_1, \cdots, P^i_c)$ indicates that the probability of the image containing the j-th category is P^i_j.

However, we do not have the ground truth labels for each proposal. Thus probability distribution can not be directly learned. Fortunately, in the multi-label image annotation, there is a label for the whole image, e.g., (I, Y), where I represents an image and Y is the ground truth label. $Y \in \mathbb{R}^c$ and $Y_j \in \{1, 0\}, j = 1, \cdots, c$. Y_j is equal to 1 if the j-th label is relevant to image I and 0 for the irrelevant case. Hence, we can firstly fuse the N proposals into one and then use the whole image's label to learn as shown in Figure 3.

More specifically, with the $N \times d$ matrix $\mathbf{D}$ in which the i-th row $\mathbf{D}^i$ represents the d-dimensional intermediate feature for the i-th proposal, in this module, we first use a fully-connected layer to compress $\mathbf{D}^i$ to a c-dimensional vector $\mathbf{M}^i \in \mathbb{R}^c$ $(i = 1, 2, \ldots, N)$.

[2]http://dl.caffe.berkeleyvision.org/bvlc_googlenet.caffemodel

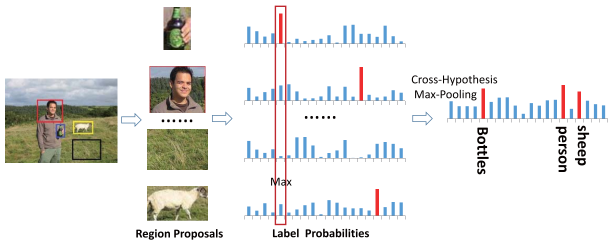


Fig. 3. Illustration of the proposed label probability calculation module. For an image (in c classes) with N region proposals, our network generates a probability vector for each proposal, e.g., $M^i \in \mathbb{R}^c$ $(i = 1, \cdots, N)$. After that, the cross-hypothesis max-pooling is used to fuse the N probability vectors into one vector.

After that, we use the cross-hypothesis max-pooling [29] to fuse $\mathbf{M}^1, \mathbf{M}^2, \ldots, \mathbf{M}^N$ to one c-dimensional vector. Specifically, let $\mathbf{M}$ be the N by c matrix whose i-th row is $\mathbf{M}^i$. The cross-hypothesis max-pooling can be formulated as

$$\mathbf{m}_j = \max\{\mathbf{M}^1_j, \mathbf{M}^2_j, \cdots, \mathbf{M}^N_j\}, \quad \forall j = 1, \cdots, c, \tag{1}$$

where $\mathbf{m}_j$ is the pooled value that corresponds to the j-th category.

Using $\mathbf{m}_j$ $(j = 1, 2, \ldots, c)$, we calculate a probability distribution $\mathbf{p} = (\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_c)$ expressed by

$$\mathbf{p}_j = \frac{\exp(\mathbf{m}_j)}{\sum_{k=1}^{c} \exp(\mathbf{m}_k)}, \tag{2}$$

where $\mathbf{p}_j$ can be regarded as the probability score that the input image contains an object in the j-th category. Using such cross-hypothesis max-pooling, if the i-th proposal contains the j-th category, then the output $\mathbf{p}_j$ should have a large value and $\mathbf{M}^i_j$ will have a high response. Hence, it can guide the learning of $\mathbf{M}$.

In this module, we define a loss function based on the cross entropy between the probability scores and the ground truth labels:

$$\ell_C = -\sum_{j \in c_+} \frac{1}{|c_+|} \log(\mathbf{p}_j), \tag{3}$$

where we denote c_+ as the set of categories which the input image belongs to, and $|c_+|$ as the number of elements in c_+. This loss function is also referred to as Softmax-Loss [30], which is a widely used loss function in the Convolutional neural networks. The (sub-)gradients with respect to $\mathbf{m}_j$ are

$$\frac{\partial \ell_C}{\partial \mathbf{m}_j} = \begin{cases} \mathbf{p}_j - \dfrac{1}{|c_+|}, & if \ Y_j = 1 \\ \mathbf{p}_j, & if \ Y_j = 0. \end{cases} \tag{4}$$

It can be easily integrated in back propagation in neural networks.

After that, similarly to $\mathbf{p}$, we can define a probability matrix $\mathbf{P}$ for the region proposals as

$$\mathbf{P}^i_j = \frac{\exp(\mathbf{M}^i_j)}{\sum_{k=1}^{c} \exp(\mathbf{M}^i_k)} (i = 1, \cdots, N), \tag{5}$$

where $\mathbf{P}^i_j$ represents the j-th element in the i-th row of $\mathbf{P}$. $\mathbf{P}^i_j$ can be viewed as the probability that the i-th proposal contains an object of the j-th category.

D. Hash Coding Module

In this subsection, we will show how to convert the image representation into (a) one piece of hash code for semantic hashing or (b) multiple pieces of hash codes, each piece corresponding to a category, for category-aware hashing.

With the $N \times d$ matrix $\mathbf{D}$ as the input, in this module, we first use a fully-connected layer to compress each $\mathbf{D}^i$ to a b-dimensional vector $\mathbf{H}^i \in \mathbb{R}^b$ $(i = 1, 2, \dots, N)$, where $\mathbf{H}^i$ corresponds to the i-th proposal. We denote $\mathbf{H}$ as the N by b matrix whose i-th row is $\mathbf{H}^i$.

1) Cross-Proposal Fusion: In order to convert $\mathbf{H}$ into the instance-aware representation of the input image, we propose a *cross-proposal fusion* strategy, by using the probability matrix $\mathbf{P}$ from the label probability calculation module.

Specifically, with the N by b feature matrix $\mathbf{H}$ and the N by c matrix $\mathbf{P}$ where $\mathbf{P}^i_j$ represents the probability of the i-th region proposal belonging to the j-th category, we fuse $\mathbf{H}$ and $\mathbf{P}$ into a long vector with $c \times b$ elements. This vector is organized in c groups, each group representing b-dimensional features corresponding to one category.

Let $\mathbf{H}^i$, $\mathbf{P}^i$ represent the i-th row of $\mathbf{H}$, $\mathbf{P}$, respectively. The cross-proposal fusion can be formulated as

$$\mathbf{f} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{P}^i \otimes \mathbf{H}^i, \tag{6}$$

where $\otimes$ is the Kronecker product. For the c-dimensional vector $\mathbf{P}^i$ and the b-dimensional vector $\mathbf{H}^i$, the Kronecker product $\mathbf{P}^i \otimes \mathbf{H}^i$ is a $(c \times b)$-dimensional vector:

$$\begin{aligned}(&\mathbf{P}^i_1\mathbf{H}^i_1, \mathbf{P}^i_1\mathbf{H}^i_2, \dots, \mathbf{P}^i_1\mathbf{H}^i_b,\\ &\mathbf{P}^i_2\mathbf{H}^i_1, \mathbf{P}^i_2\mathbf{H}^i_2, \dots, \mathbf{P}^i_2\mathbf{H}^i_b,\\ &\dots\\ &\mathbf{P}^i_c\mathbf{H}^i_1, \mathbf{P}^i_c\mathbf{H}^i_2, \dots, \mathbf{P}^i_c\mathbf{H}^i_b).\end{aligned}$$

Let $\mathbf{f} = (\mathbf{f}^{(1)}, \mathbf{f}^{(2)}, \dots, \mathbf{f}^{(c)})$, where $\mathbf{f}^{(j)}$ is a b-dimensional vector. It is easy to verify that

$$\mathbf{f}^{(j)} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{P}^i_j \mathbf{H}^i.$$

Since $\mathbf{H}^i$ represents the features of the i-th proposal, $\mathbf{f}^{(j)}$ can be regarded as the weighted average of the proposals' features. If the i-th proposal has a relatively higher/lower score $\mathbf{P}^i_j$ (meaning that the i-th proposal likely/unlikely belongs to the j-th category), the feature vector $\mathbf{H}^i$ (associated to the i-th proposal) has more/less contribution to the weighted average $\mathbf{f}^{(j)}$.

Figure 4 shows an illustrative example of cross-proposal fusion. Suppose there are only 2 categories (*bicycle* and *sofa*, $c = 2$) in all of the images. For the input image, 4 region proposals are generated in the first module ($N = 4$). Then, suppose the label probability calculation module generates a 4×2 probability matrix $\mathbf{P}$ with $\mathbf{P}^1 = (0.99, 0.01)$, $\mathbf{P}^2 = (0.9, 0.1)$, $\mathbf{P}^3 = (0.05, 0.95)$ and $\mathbf{P}^4 = (0.08, 0.92)$. For the 1st proposal, $\mathbf{P}^1 = (0.99, 0.01)$ indicates that it is very likely to contain a bicycle (with a score 0.99), but it seems unlikely to contain a sofa (with a score 0.01). In the

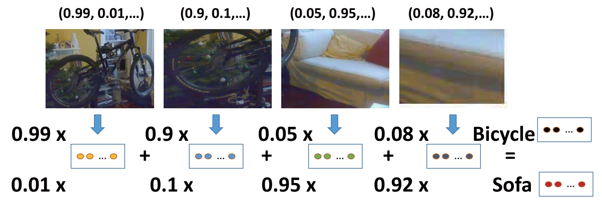


Fig. 4. Illustration of the cross-proposal fusion. Each region proposal is firstly encoded into a feature vector. And these feature vectors are fused into an intermediate feature representation, by using the probability scores (learned from the label probability calculation module) as the weights.

hash coding module, the i-th proposal is represented by a b-dimensional feature vector $\mathbf{H}^i$. Finally, for the input image, we conduct cross-proposal fusion to obtain an instance-aware representation $\mathbf{f} = (\mathbf{f}^{(1)}, \mathbf{f}^{(2)})$, where the representation of "*bicycle*" is $\mathbf{f}^{(1)} = \frac{1}{4}(0.99\mathbf{H}^1 + 0.9\mathbf{H}^2 + 0.05\mathbf{H}^3 + 0.08\mathbf{H}^4)$, and the representation of "*sofa*" is $\mathbf{f}^{(2)} = \frac{1}{4}(0.01\mathbf{H}^1 + 0.1\mathbf{H}^2 + 0.95\mathbf{H}^3 + 0.92\mathbf{H}^4)$.

The cross-proposal fusion is a crucial step for the instance-aware image representation. If the image contains an object, then the instance-aware representation will have a high response for the object. It also gives us a simple way to combine the multi-label information into the hashing procedure.

Discussions: A concern arising here is that some input proposals may be inaccurate or even do not contain any objects, which will make the features generated by these proposals noisy and harm the final performance. We argue that, to some extent, the operations in the proposed Cross-Hypothesis Max-Pooling and the Cross-Proposal Fusion can reduce the negative effects of the possibly noisy input proposals. Firstly, in the label probability calculation module before the cross-hypothesis max-pooling, each input proposal is assigned with a set of probability scores (one score for one label). Higher scores are assigned to the proposals that may contain objects with more confidence, and lower scores are assigned to those noisy proposals. Hence, those noisy proposals may more probably be suppressed by the cross-hypothesis max-pooling. Secondly, similar to [31], in the cross-proposal fusion, the proposal' features are weighted by their probability scores. Hence, those noisy proposals' features have less contribution to the final feature representation. In summary, the re-weighting operations in the cross-hypothesis max-pooling and the cross-proposal fusion can reduce the negative effects of the inaccurate input proposals.

With $\mathbf{f}$ generated by the cross-proposals fusion, we can generate either the category-aware hash representation that consists of c pieces of hash codes, or the semantic hash representation that consists of one piece of hash code. Next we will present these cases separately.

2) Category-Aware Hash Representation: Since $\mathbf{f}$ is organized in c groups $\mathbf{f}^{(1)}, \mathbf{f}^{(2)}, \dots, \mathbf{f}^{(c)}$, each $\mathbf{f}^{(i)} (i = 1, 2, \dots c)$ can be converted into a b-bit binary code $\mathbf{b}^{(i)} = \text{sign}(\mathbf{f}^{(i)})$, where $\text{sign}(x) = 1$ if $x > 0$, and otherwise $\text{sign}(x) = 0$. For an image I, the category-aware hash representation of I is $\mathbf{b}(I) = (\mathbf{b}^{(1)}(I), \mathbf{b}^{(2)}(I), \dots, \mathbf{b}^{(c)}(I))$.

To learn this representation, we define c triplet loss functions [8], [32], each for one category. To obtain triplet samples, we randomly select image I^+ and image I that belong to the same category and the negative image I^- is randomly selected from those which do not contain the category. Then we design a triplet loss that tries to preserve the relative similarities in the form: "image I is more similar to image I^+ than to I^-". For the j-th category ($j = 1, 2, \dots, c$), suppose we have three images I, I^+ and I^-, where both I and I^+ belong to the j-th category, but I^- does not. Then the triplet loss associated to the j-th category is defined by

$$\begin{aligned}&\ell_{Triplet}(\mathbf{f}^{(j)}(I), \mathbf{f}^{(j)}(I^+), \mathbf{f}^{(j)}(I^-))\\&\quad = \max(0, 1 - \|\mathbf{f}^{(j)}(I) - \mathbf{f}^{(j)}(I^-)\|_2^2\\&\qquad + \|\mathbf{f}^{(j)}(I) - \mathbf{f}^{(j)}(I^+)\|_2^2)),\end{aligned} \tag{7}$$

where $\mathbf{f}^{(j)}(I)$ represents the vector $\mathbf{f}^{(j)}$ for the image I.

This loss function is convex. Its sub-gradient with respect to $\mathbf{f}^{(j)}(I)$, $\mathbf{f}^{(j)}(I^+)$ and $\mathbf{f}^{(j)}(I^-)$ can be easily obtain by

$$\begin{aligned}&\frac{\partial \ell_{Triplet}(\mathbf{f}^{(j)}(I), \mathbf{f}^{(j)}(I^+), \mathbf{f}^{(j)}(I^-))}{\mathbf{f}^{(j)}(I)}\\&\quad = 2(\mathbf{f}^{(j)}(I^-) - \mathbf{f}^{(j)}(I^+))\\&\frac{\partial \ell_{Triplet}(\mathbf{f}^{(j)}(I^+), \mathbf{f}^{(j)}(I^+), \mathbf{f}^{(j)}(I^-))}{\mathbf{f}^{(j)}(I^+)}\\&\quad = 2(\mathbf{f}^{(j)}(I^+) - \mathbf{f}^{(j)}(I))\\&\frac{\partial \ell_{Triplet}(\mathbf{f}^{(j)}(I^-), \mathbf{f}^{(j)}(I^+), \mathbf{f}^{(j)}(I^-))}{\mathbf{f}^{(j)}(I^-)}\\&\quad = 2(\mathbf{f}^{(j)}(I) - \mathbf{f}^{(j)}(I^-))\end{aligned} \tag{8}$$

when $1 - \|\mathbf{f}^{(j)}(I) - \mathbf{f}^{(j)}(I^-)\|_2^2 + \|\mathbf{f}^{(j)}(I) - \mathbf{f}^{(j)}(I^+)\|_2^2 > 0$. Otherwise the sub-gradients are all zeros.

3) Semantic Hash Representation: For semantic hashing, we assume the target length of a hash code is q bits. We first use a fully-connected layer to convert the $(c \times b)$-dimensional $\mathbf{f}$ to a q-dimensional $\mathbf{s}$. $\mathbf{s}$ can be converted to a q-bit binary code by sign($\mathbf{s}$), where $\text{sign}(x) = 1$ if $x > 0$, and otherwise $\text{sign}(x) = 0$.

Next we present the triplet loss defined on $\mathbf{s}$. Since the original triplet loss [8], [32] is designed for single-label data, here we propose a weighted triplet loss for multi-label data. Specifically, we define the similarity function $sim(I_a, I_b)$ as the number of shared labels between the images I_a and I_b. Then, for the images I, I^+ and I^- and $sim(I, I^+) > sim(I, I^-)$, the weighted triplet loss is defined by

$$\begin{aligned}&\ell_{W-Triplet}(\mathbf{s}(I), \mathbf{s}(I^+), \mathbf{s}(I^-))\\&\quad = (2^{sim(I,I^+)} - 2^{sim(I,I^-)})\ell_{Triplet}(\mathbf{s}(I), \mathbf{s}(I^+), \mathbf{s}(I^-))\end{aligned} \tag{9}$$

where $\ell_{Triplet}$ is defined in (7), and $\mathbf{s}(I)$ is the q-dimensional vector for the image I.

In many existing supervised hashing methods, the side information is in the form of pairwise labels indicating the similarites/dissimilarites on image pairs. In these hashing methods, a straightforward way is to define the pairwise loss functions which preserve the pairwise similarities of images. Some recent papers (e.g., [8], [32]) learn hash functions by using triplet loss functions, which seek to preserve the relative similarities in the form: "image A is more similar to image B than to image C". Such a form of triplet-based relative similarities can be more easily obtained than pairwise similarities (e.g., users' click-through data from image retrieval applications).

IV. Category-Aware Retrieval

Suppose that we have a set of images $\mathcal{S} = \{I_1, I_2, \dots, I_{|\mathcal{S}|}\}$ with $|\mathcal{S}|$ being the number of images in $\mathcal{S}$ for retrieval. Independently from other categories, for the j-th category ($j = 1, 2, \dots, c$), we can generate the binary codes $\mathbf{b}^{(j)}(I_1), \mathbf{b}^{(j)}(I_2), \dots, \mathbf{b}^{(j)}(I_{|\mathcal{S}|})$, and then conduct retrieval based on these codes. Hence, for a query image, the retrieved results can be organized in c groups, where the j-th group has a list of images, each of which is likely to contain a similar object of the j-th category.

An issue which needs to be considered here is that the number of objects in an image may be less than c, and it is inappropriate to organize the retrieved results in c groups for all of the query images. Since the label probability calculation module (see Section III-C) outputs a probability vector $\mathbf{p}$, where $\mathbf{p}_j$ represents the predicted value for the possibility of the input image containing objects in the j-th category. For the c groups of retrieved results, we can remove the j-th group if and only if $\mathbf{p}_j$ is less than some threshold. In our experiments, we empirically set this threshold to be 0.2.

For those images in the database for retrieval, each image is first encoded into c pieces of b-bit binary codes. Then we collect those hash codes with a probability score (i.e., $\mathbf{p}_j$ for the j-th piece of hash code) no less than 0.2. We organize the collected hash codes in c groups. The j-th group contains the hash codes with each being the j-th piece of code of some image. Finally, we build a hash table to store the hash codes in each group, respectively. In retrieval, for a test query image, we first convert it into c pieces of b-bit codes, and then remove those codes with a probability score less than 0.2. For each of the rest hash codes, we conduct search in the corresponding hash table and obtain a list of retrieved images.

Figure 5 shows two examples of results from our experiments. For the first example, when retrieving with a query image containing a *cat* and a *dog*, the proposed method returns two lists of retrieved images. Each image in the first/second list is likely to have a *cat*/*dog* similar to that in the query image, where the approximate location of this *cat*/*dog* is also indicated (in the associated grey image). The grey images are saliency maps that are obtained in favor of the automatically generated region proposals (see Section III-A) and the predicted probability scores (i.e., the vector $\mathbf{P}$ in Section III-C).

V. Experiments

In this section, we evaluate the performance of the proposed method for either semantic hashing or category-aware hashing, and compare it with several state-of-the-art hashing methods.

A. Datasets and Evaluation Metrics

We evaluate the proposed method on three public datasets of multi-label images: VOC 2007 [33], VOC 2012 [33] and MIRFLICKR-25K [34].

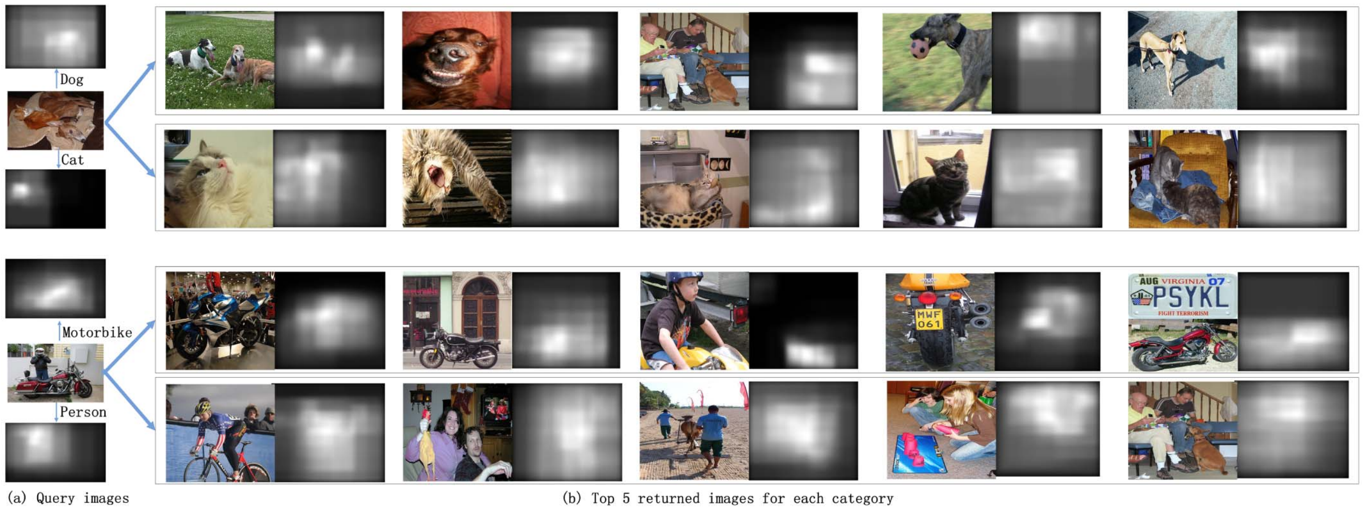


Fig. 5. Two examples of the results of category-aware hashing. Each retrieved image has a corresponding grey image of the saliency map, in which whiter color indicates higher possibility of an object existing at that position. Given a query image, the proposed method returns multiple lists of images, each list corresponding to a category. Each image in a returned list is likely to contain a similar object as in the query image, where the approximate location of this object is shown in the corresponding grey image.

TABLE I

THE NUMBER OF TRAINING SAMPLES AND TESTING SAMPLES

	VOC2007	VOC2012	MIRFLICKR-25K
#Test	2,000	2,000	2,000
#Train	7,963	9,540	23,000

- VOC 2007 consists of 9,963 multi-label images which are collected from Flickr.[3] There are 20 object classes in this dataset. On average, each image is annotated with 1.5 labels.
- VOC 2012 consists of 22,531 multi-label images in 20 classes. Since the ground truth labels of the test images are not available, in our experiments, we only use 11,540 images from its training and validation set.
- MIRFLICKR-25K consists of 25,000 multi-label images downloaded from Flickr. There are 38 classes in this dataset. Each image has 4.7 labels on average.

In each dataset, we randomly select 2,000 images as the test query set, and the rest images are used as training samples. Note that, we only use 11,540 images in VOC2012 dataset, which have the ground truth labels. The number of training samples and testing samples are shown in Table I:

To evaluate the performance, we use four evaluation metrics: Normalized Discounted Cumulative Gains (NDCG) [35], Mean Average Precision (MAP) [36], Weighted MAP [9] and Average Cumulative Gains (ACG) [37].

NDCG is a popular evaluation metric in information retrieval. Given a query image, the DCG score at the position m is defined as

$$\text{DCG@}m = \sum_{j=1}^{m} \frac{2^{r(j)} - 1}{\log(1+j)}, \tag{10}$$

where $r(j)$ is the similarity between the j-th position image and the query image, which is defined as the number of shared labels between the query image and the j-th retrieved image. Then, the NDCG score at the position m can be calculated by $\text{NDCG@}m = \frac{\text{DCG@}m}{Z_m}$, where Z_m is the maximum value of DCG@m, making the value of NDCG fall in the range [0, 1].

ACG@m represents the sum of similarities between the query image and each of the top m retrieved images, which can be calculated by

$$\text{ACG@}m = \sum_{j=1}^{m} \frac{r(j)}{m}. \tag{11}$$

MAP is a standard evaluation metric for information retrieval. It is the mean of averaged precisions over a set of queries, which can be calculated by

$$\text{MAP} = \sum_{j=1}^{n} P@j \times pos(j)/N_{pos}, \tag{12}$$

where $pos(j)$ is an indicator function. If the image at the position j is relevant (i.e., it at least shares one label with the query image), $pos(j)$ is 1; otherwise $pos(j)$ is 0. N_{pos} represents the total number of relevant images w.r.t. the query image. $P@j = \frac{N_{pos}(j)}{j}$, where $N_{pos}(j)$ represents the number of relevant images within the top j images.

The weighted MAP is defined as

$$\text{Weighted MAP} = \sum_{j=1}^{M} \text{ACG@}j \times pos(j)/N_{pos}. \tag{13}$$

B. Experimental Setting

We implement the proposed method based on the open-source *Caffe* [38] framework. The networks are trained using stochastic gradient descent. In training, the weights of the layers are initialized by the pre-trained GoogLeNet model. The base learning rate is set to be 0.0001. After every 30 epochs on the training data, the learning rate is adjusted to one tenth

[3]http://www.flickr.com/

TABLE II
COMPARISON RESULTS OF HAMMING RANKING W.R.T. DIFFERENT NUMBERS OF BITS ON THREE DATASETS

Methods	VOC 2007				MIRFLICKR25K				VOC 2012			
	16 bits	32 bits	48 bits	64 bits	16 bits	32 bits	48 bits	64 bits	16 bits	32 bits	48 bits	64 bits
NDCG@1000												
Ours	**0.7963**	**0.8696**	**0.8865**	**0.8929**	**0.4725**	**0.5245**	**0.5400**	**0.5552**	**0.7939**	**0.8385**	**0.8566**	**0.8573**
One-Stage	0.7808	0.8296	0.8446	0.8530	0.4413	0.5096	0.5392	0.5550	0.7540	0.8012	0.8170	0.8224
ITQ-CCA	0.7704	0.8007	0.8139	0.8146	0.4498	0.4719	0.4866	0.4921	0.7471	0.7759	0.7815	0.7891
ITQ	0.6848	0.6768	0.6783	0.6766	0.3734	0.3934	0.3966	0.3982	0.6433	0.6371	0.6338	0.6337
SH	0.5404	0.5013	0.4796	0.4697	0.3096	0.3046	0.2998	0.2959	0.5157	0.4718	0.4409	0.4238
ACG@1000												
Ours	**0.7065**	**0.7590**	**0.7674**	**0.7731**	**2.6751**	**2.8353**	2.8951	2.9282	**0.7523**	**0.7841**	**0.7972**	**0.7963**
One-Stage	0.7007	0.7323	0.7407	0.7483	2.6083	2.8302	**2.9141**	**2.9642**	0.7141	0.7522	0.7654	0.7705
ITQ-CCA	0.6418	0.6658	0.6780	0.6779	2.4785	2.5314	2.5964	2.6149	0.6733	0.6971	0.7031	0.7107
ITQ	0.5823	0.5695	0.5676	0.5661	2.1964	2.2568	2.2650	2.2747	0.5794	0.5715	0.5669	0.5656
SH	0.4570	0.4218	0.4044	0.3982	1.8394	1.7668	1.7215	1.6894	0.4660	0.4204	0.3947	0.3785
MAP												
Ours	**0.7997**	**0.8618**	**0.8784**	**0.8830**	**0.7994**	**0.8317**	**0.8366**	**0.8361**	**0.7942**	**0.8437**	**0.8617**	**0.8642**
One-Stage	0.7488	0.7995	0.8171	0.8259	0.7727	0.8059	0.8136	0.8179	0.7343	0.7870	0.8055	0.8109
ITQ-CCA	0.6913	0.7264	0.7404	0.7396	0.7015	0.7053	0.7174	0.7254	0.6952	0.7254	0.7362	0.7427
ITQ	0.5845	0.5747	0.5769	0.5741	0.6804	0.6822	0.6796	0.6795	0.5715	0.5984	0.5554	0.5549
SH	0.4432	0.4071	0.3875	0.3799	0.6174	0.6057	0.5994	0.5952	0.4378	0.4184	0.3641	0.3485
Weighted MAP												
Ours	**0.8566**	**0.9255**	**0.9449**	**0.9505**	**2.0877**	**2.1926**	**2.2271**	**2.2294**	**0.8429**	**0.9005**	**0.9205**	**0.9229**
One-Stage	0.8007	0.8595	0.8794	0.8903	2.0411	2.1584	2.1958	2.2187	0.7798	0.8414	0.8631	0.8698
ITQ-CCA	0.7325	0.7725	0.7879	0.7866	1.7359	1.7518	1.7982	1.8185	0.7312	0.7666	0.7779	0.7854
ITQ	0.6214	0.6129	0.6163	0.6132	1.6269	1.6403	1.6344	1.6369	0.6051	0.5715	0.5911	0.5906
SH	0.4723	0.4342	0.4137	0.4051	1.4077	1.3598	1.3324	1.3150	0.4637	0.4205	0.3865	0.3697

of the current learning rate. In all of our experiments, we first use GOP to obtain the bounding boxes of region proposals (no more than 1000 proposals for an image). With these bounding boxes, we use non-maximum suppression to obtain a smaller number of boxes, and then select the top N (here we set $N = 100$) boxes with the highest confidence. We use the 4-level pyramid pooling ($4\times4, 3\times3, 2\times2, 1\times1$). The number of feature maps in the last convolution layer is 32, hence the dimension d of each intermediate feature vector is 960 (i.e., $32 \times (4 \times 4 + 3 \times 3 + 2 \times 2 + 1 \times 1)$). The number b of a proposal's feature vectors is set to be the desired hash bits for each category.

During training, we use a randomly sampling strategy to generate triplets (i.e., a triplet (I, I^+, I^-) of images that I is more similar to I^+ than to I^-). Specifically, the proposed network is trained by stochastic gradient descent, where the number of iterations is 15,000 and the mini-batch size is 32. We denote *SharedLabels*(I_1, I_2) [4] as the number of shared labels between the image I_1 and the image I_2. The procedure of generating triplets in each iteration is shown in Algorithm 1.

The source code of the proposed method is made publicly available at http://ss.sysu.edu.cn/~py/tip-hashing.rar.

Algorithm 1 Procedure of Generating Triplets

```
Input: a batch $S$ of 32 training images.
Output: a set $T$ of triplets.
$T \leftarrow \emptyset$.
For every triplet $t = (I_1, I_2, I_3)$ that $I_1 \in S, I_2 \in S$ and $I_3 \in S$
    If $SharedLabels(I_1, I_2) > SharedLabels(I_1, I_3)$
        $T \leftarrow T \bigcup t$
    End If
End For
Output $T$.
```

C. Results on Semantic Hashing

The first set of experiments is to evaluate the performance of the proposed method in semantic hashing.

We use SH [14], ITO [1], ITQ-CCA [1] and One-Stage Hashing [8] as the baselines in our experiments. SH and ITQ are unsupervised methods, while ITQ-CCA and one-stage hashing are supervised methods. One-Stage hashing is a recently proposed deep-networks-based hashing method that is the most related competitor to the proposed method. For SH, ITQ and ITQ-CCA, we use the pre-trained GoogLeNet model[5] to extract features for the images. The feature vector for each image is with 1024 dimensions. For a fair comparison, in our implementation of One-Stage Hashing, we use the architecture of GoogLeNet as its shared sub-network, instead of the NIN architecture used in [8]; we also use the same weighted triplet loss in (9) as the proposed method. The variant of One-Stage Hashing also uses the open-source *Caffe* for training.

Table II shows the comparison results w.r.t. NDCG@1000, ACG@1000, MAP and Weighted MAP. Figure 6 and Figure 7 show the NCCG@m and ACG@m with varying m. As can be seen, the proposed method shows superior performance gains over the baselines. On VOC 2007 and VOC 2012, the NDCG@1000 values of the proposed methods indicate a 1.9% $\sim$ 4.9% / 4.2% $\sim$ 5.2% relative increase over the second best baseline. The ACG@1000 value of the proposed method is 0.7731 with 64 bits, compared to 0.7483 of

[4]As an illustrative example, suppose I_1 has the class labels a, b, and c, I_2 has the class labels a, c, and d, then we have *SharedLabels*$(I_1, I_2) = 2$ because I_1 and I_2 have shared labels a and c.

[5]https://github.com/BVLC/caffe/tree/master/models/bvlc_googlenet

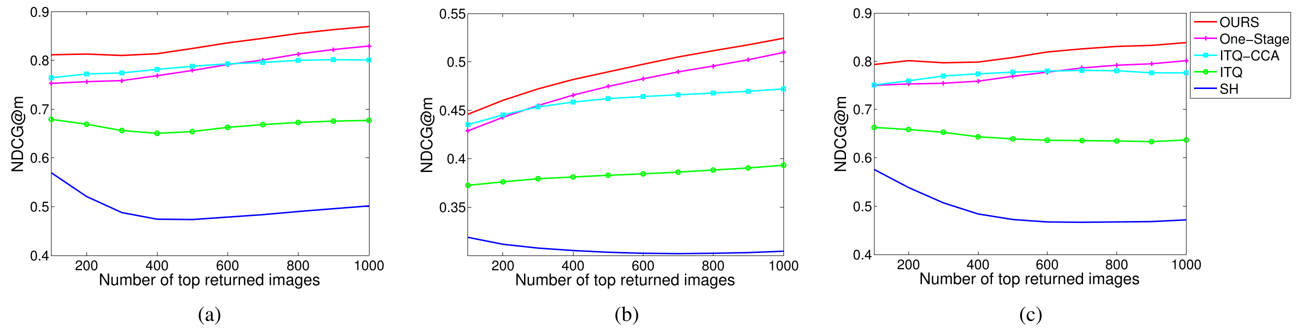


Fig. 6. NDCG curves with 32 bits w.r.t. different numbers of top returned samples. (a) VOC 2007. (b) MIRFLICKR. (c) VOC 2012.

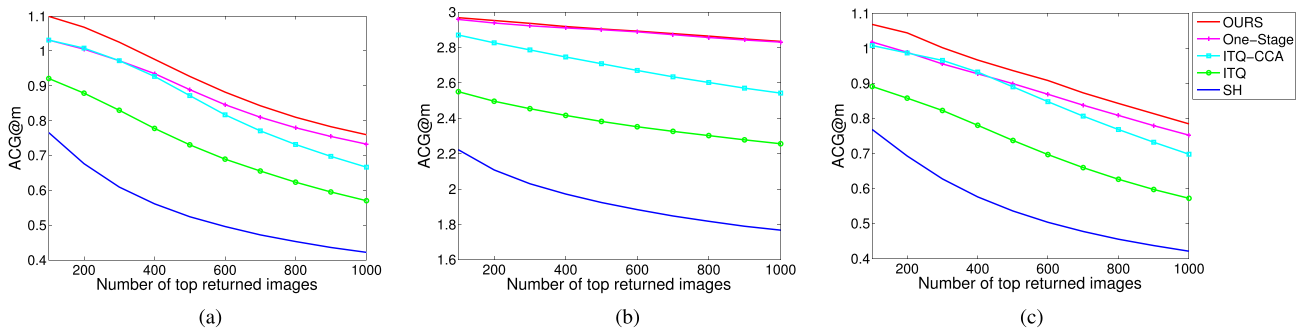


Fig. 7. ACG curves with 32 bits w.r.t. different numbers of top returned samples. (a) VOC 2007. (b) MIRFLICKR. (c) VOC 2012.

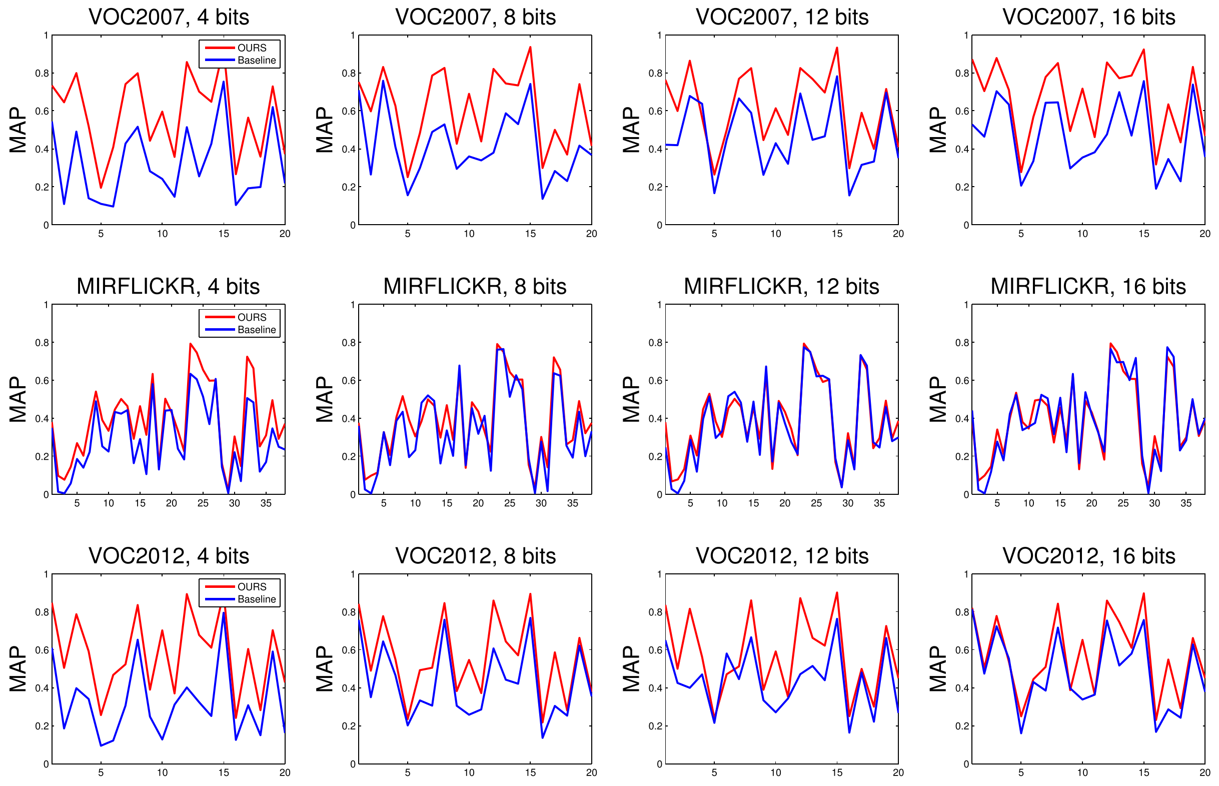


Fig. 8. Per-category MAP curves on three datasets. The x-axis represents the categories.

One-Stage Hashing. On MIRFLICKR-25K, the values of MAP indicate a relative increase of 2.2% $\sim$ 3.4%. It can be observed from these results that incorporating automatically generated region proposals and label probability calculation in the process of hash learning can help improve the performance of semantic hashing.

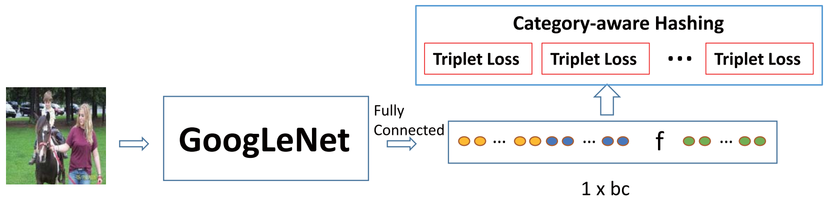


Fig. 9. The architecture of the baseline for category-aware hashing.

D. Results on Category-Aware Hashing

We also evaluate the performance of the proposed method for category-aware hashing. Since little effort has been devoted to category-aware hashing on multi-label images, to demonstrate the advantages of the proposed method, we implement a deep-networks-based baseline that also outputs c pieces of b-bit hash codes, each code corresponding to a category. As shown in Figure 9, this baseline adopts GoogLeNet as the basic framework. After the last (1024D) fully connected layer of GoogLeNet, a fully connected layer with $c \times b$ nodes is added, and then this layer is separated into c slices (each is in b dimensions). For the j-th ($j = 1, 2, \ldots, c$) slice, a triplet loss is defined which regards the images belonging to the j-th category as positive examples, and other images as negative ones. To train this baseline, we also use the pre-trained GoogLeNet model to initialize its weights.

The baseline is a simpler category-aware retrieval system, which does not use the region proposal module and label probability module. The experimental results can answer us whether the retrieval system with these two modules can contribute to the accuracy improvement or not.

For a test query image, we first convert it into c pieces of b-bit codes, and then use the hash codes of categories that the test image contains to conduct search in the corresponding hash table and obtain a list of retrieved images.

The MAP results (for each category) are shown in Figure 8. We can observe that the proposed method consistently outperforms the baseline. For example, on VOC 2007 with $b = 4$, the averaged MAP (over 20 classes) of the proposed method is 0.5831, compared to 0.3190 of the baseline. On VOC 2012, the averaged MAP of the proposed method has a relative increase of 78.64% over the baseline with $b = 12$. On MIRFILCKR-25K, the proposed method yields a 12.89% relative increase over the baseline with $b = 8$ w.r.t. averaged MAP. Figure 5 shows two examples of results from our experiments.

VI. Conclusions and Future Work

In this paper, we proposed a deep-networks-based hashing method for multi-label image retrieval, by incorporating automatically generated region proposals and label probability calculation in the hash learning process. In the proposed deep architecture, an input image is converted to an instance-aware representation organized in groups, each group corresponding to a category. Based on this representation, we can easily generate binary hash codes for either semantic hashing or category-aware hashing. Empirical evaluations on both the category-aware hashing and semantic hashing show that the proposed method substantially outperforms the state-of-the-arts.

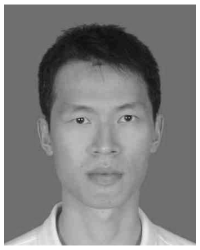
In future work, we plan to study unsupervised instance-aware image retrieval, in which the virtual classes can be obtained by clustering.

References

[1] Y. Gong and S. Lazebnik, "Iterative quantization: A procrustean approach to learning binary codes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2011, pp. 817–824.
[2] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search," in *Proc. IEEE Int. Conf. Comput. Vis.*, Sep./Oct. 2009, pp. 2130–2137.
[3] W. Liu, J. Wang, S. Kumar, and S.-F. Chang, "Hashing with graphs," in *Proc. Int. Conf. Mach. Learn.*, 2011, pp. 1–8.
[4] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, "Supervised hashing for image retrieval via image representation learning," in *Proc. AAAI Conf. Artif. Intell.*, 2014, pp. 2156–2162.
[5] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 2074–2081.
[6] B. Kulis and T. Darrell, "Learning to hash with binary reconstructive embeddings," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 1042–1050.
[7] G. Lin, C. Shen, D. Suter, and A. van den Hengel, "A general two-step approach to learning-based hashing," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 2552–2559.
[8] H. Lai, Y. Pan, Y. Liu, and S. Yan, "Simultaneous feature learning and hash coding with deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 3270–3278.
[9] F. Zhao, Y. Huang, L. Wang, and T. Tan, "Deep semantic ranking based hashing for multi-label image retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1556–1564.
[10] R. Salakhutdinov and G. Hinton, "Semantic hashing," *Int. J. Approx. Reasoning*, vol. 50, no. 7, pp. 969–978, Jul. 2009.
[11] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 346–361.
[12] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proc. Int. Conf. Very Large Data Bases*, 1999, pp. 518–529.
[13] R. Salakhutdinov and G. E. Hinton, "Learning a nonlinear embedding by preserving class neighbourhood structure," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2007, pp. 412–419.
[14] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 1753–1760.
[15] M. Norouzi and D. M. Blei, "Minimal loss hashing for compact binary codes," in *Proc. Int. Conf. Mach. Learn.*, 2011, pp. 353–360.
[16] X. Li, G. Lin, C. Shen, A. van den Hengel, and A. Dick, "Learning hash functions using column generation," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 142–150.
[17] J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for scalable image retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 3424–3431.
[18] D. Tao, X. Tang, X. Li, and X. Wu, "Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 7, pp. 1088–1099, Jul. 2006.
[19] A. Torralba, R. Fergus, and Y. Weiss, "Small codes and large image databases for recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 1–8.

[20] K. Lin, H.-F. Yang, J.-H. Hsiao, and C.-S. Chen, "Deep learning of binary hash codes for fast image retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2015, pp. 27–35.
[21] D. Wang, P. Cui, M. Ou, and W. Zhu, "Deep multimodal hashing with orthogonal regularization," in *Proc. 24th Int. Joint Conf. Artif. Intell.*, 2015, pp. 2291–2297.
[22] J. Carreira and C. Sminchisescu, "CPMC: Automatic object segmentation using constrained parametric min-cuts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 7, pp. 1312–1328, Jul. 2012.
[23] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *Int. J. Comput. Vis.*, vol. 104, no. 2, pp. 154–171, Apr. 2013.
[24] P. Arbeláez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik, "Multi-scale combinatorial grouping," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 328–335.
[25] M.-M. Cheng, Z. Zhang, W.-Y. Lin, and P. Torr, "BING: Binarized normed gradients for objectness estimation at 300 fps," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 3286–3293.
[26] P. Krähenbühl and V. Koltun, "Geodesic object proposals," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 725–739.
[27] C. Szegedy *et al.* (2014). "Going deeper with convolutions." [Online]. Available: http://arxiv.org/abs/1409.4842
[28] D. Erhan, P.-A. Manzagol, Y. Bengio, S. Bengio, and P. Vincent, "The difficulty of training deep architectures and the effect of unsupervised pre-training," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2009, pp. 153–160.
[29] Y. Wei *et al.* (2014). "CNN: Single-label to multi-label." [Online]. Available: http://arxiv.org/abs/1406.5726
[30] Y. Gong, Y. Jia, T. Leung, A. Toshev, and S. Ioffe. (2013). "Deep convolutional ranking for multilabel image annotation." [Online]. Available: http://arxiv.org/abs/1312.4894
[31] T. Liu and D. Tao. (2014). "Classification with noisy labels by importance reweighting." [Online]. Available: http://arxiv.org/abs/1411.7718
[32] M. Norouzi, D. J. Fleet, and R. Salakhutdinov, "Hamming distance metric learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1061–1069.
[33] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Sep. 2009.
[34] M. J. Huiskes and M. S. Lew, "The MIR flickr retrieval evaluation," in *Proc. 1st ACM Int. Conf. Multimedia Inf. Retr.*, 2008, pp. 39–43.
[35] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of IR techniques," *ACM Trans. Inf. Syst.*, vol. 20, no. 4, pp. 422–446, Oct. 2002.
[36] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*, vol. 463. New York, NY, USA: ACM Press, 1999.
[37] K. Järvelin and J. Kekäläinen, "IR evaluation methods for retrieving highly relevant documents," in *Proc. 23rd Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2000, pp. 41–48.
[38] Y. Jia. (2013). *Caffe: An Open Source Convolutional Architecture for Fast Feature Embedding*. [Online]. Available: http://caffe.berkeleyvision.org

Hanjiang Lai received the B.S. and Ph.D. degrees from Sun Yat-sen University, in 2009 and 2014, respectively. He was a Research Fellow with the National University of Singapore from 2014 to 2015. He is currently with Sun Yat-sen University. His research interests include machine learning algorithms, deep learning, and computer vision.

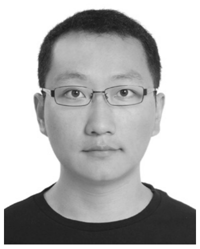


Pan Yan received the B.S. degree in information science and the Ph.D. degree in computer science from Sun Yat-sen University, Guangzhou, China, in 2002 and 2007, respectively.

He is currently an Associate Professor with Sun Yat-sen University. His current research interests include machine learning algorithms, learning to rank, and computer vision.

Dr. Pan has served as a Reviewer for several conferences and journals. He was the winner of the object categorization task in PASCAL Visual Object Classes Challenge in 2012.

Xiangbo Shu is currently pursuing the Ph.D. degree with School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. Since 2014, she has been a Visiting Scholar with the Department of Electrical and Computer Engineering, National University of Singapore. Her research interests include social multimedia mining, computer vision, and machine learning.



Yunchao Wei is currently pursuing the Ph.D. degree with the Institute of Information Science, Beijing Jiaotong University, China. He is currently with the National University of Singapore as a Research Intern. His research interests mainly include object classification in computer vision and multi-modal analysis in multimedia.

Shuicheng Yan is currently an Associate Professor with the Department of Electrical and Computer Engineering, National University of Singapore, and the Founding Lead of the Learning and Vision Research Group. His research areas include machine learning, computer vision, and multimedia. He has authored/co-authored nearly 400 technical papers over a wide range of research topics, with Google Scholar citation over 15 000 times. He was an ISI Highly Cited Researcher 2014, and an IAPR Fellow 2014. He has served as an Associate Editor of IEEE TKDE, CVIU, and TCSVT. He received the Best Paper Awards from ACM MM'13 (Best Paper and Best Student Paper), ACM MM 12 (Best Demo), PCM'11, ACM MM 10, ICME 10, and ICIMCS'09, the Runner-Up Prize of ILSVRC'13, the winner prizes of the classification task in PASCAL VOC 2010-2012, the winner prize of the segmentation task in PASCAL VOC 2012, the honorable mention prize of the detection task in PASCAL VOC'10, the 2010 TCSVT Best Associate Editor Award, the 2010 Young Faculty Research Award, the 2011 Singapore Young Scientist Award, and the 2012 NUS Young Researcher Award.