Multimed Tools Appl
https://doi.org/10.1007/s11042-018-6210-3

Indexing of the CNN features for the large scale image search

Ruoyu Liu[1] · **Shikui Wei**[1] · **Yao Zhao**[1] · **Yi Yang**[2]

Received: 10 September 2017 / Revised: 7 April 2018 / Accepted: 23 May 2018

Abstract The convolutional neural network (CNN) features can give good description of image content, which usually represent an image with a single feature vector. Although CNN features are more compact than local descriptors, they still cannot efficiently deal with large-scale retrieval due to the linearly incremental cost of computation and storage. To address this issue, we build a simple but effective indexing framework on inverted table, which significantly decreases both search time and memory usage. First, several strategies are fully investigated to adapt inverted table to CNN features for compensating for quantization error. We use multiple assignment for the query and database images to increase the probability that relevant images are assigned to the same visual word obtained via clustering. Embedding codes are also introduced to improve retrieval accuracy by removing false matches. Second, a novel indexing framework that combines inverted table and hashing codes is proposed. This framework is faster than the reformed inverted tables with the introduced strategies. Experiment on several benchmark datasets demonstrates that our method yields faster retrieval speed compared to brute-force search. We also provide fair comparison between popular CNN features.

Keywords Convolutional neural network · Indexing · Inverted table

✉ Ruoyu Liu
12112062@bjtu.edu.cn

Shikui Wei
shkwei@bjtu.edu.cn

Yao Zhao
yzhao@bjtu.edu.cn

Yi Yang
yee.i.yang@gmail.com

[1] Beijing Jiaotong University, Beijing, China

[2] University of Technology Sydney, Ultimo, Australia

Published online: 15 June 2018

1 Introduction

Image search aims to find relevant images of a specific query from mass data [79]. The main effort focuses on improving search accuracy and search efficiency. Search accuracy is closely related to feature extraction, whereas search efficiency is mainly dependent on indexing structure.

To represent image content accurately, various feature-extraction schemes have been developed [4, 8, 21, 24, 53, 55, 70], which can be roughly separated into two groups, *i.e.*, global features [50] and local features [38]. Global features such as color histograms [21], Tamura texture features [70] and moment invariants [24], are often statistics of image color, textural or shape information, and each image is described as a single short vector. Generally, this kind of feature is compact and efficient for performing image search. However, they cannot describe image details or handle transformation like rotation and change in illumination, since they only capture low-level information. By contrast, local features, *i.e.*, SIFT [53], GLOH [55], SURF [4] and HOG [8], describe an image with a set of descriptors and are better at discriminating between contents. However, their shortcoming is that the amount of local features is huge, even when features are extracted from a small-scale dataset. When facing a large-scale dataset, image search on local features probably results in unacceptable computation and storage cost. There is another branch of global features that aggregate local descriptors of an image into a unique vector, *e.g.*, Fisher vector (FV) [61] and VLAD [11, 13, 17, 30, 51].

To speed up searching, various indexing schemes have been studied. Tree-based structures, such as R-tree [20], M-tree [7] and inverted table [68, 84, 89], generally divide the entire feature space into several non-overlapping partitions. Since only a small proportion of the database images that locate in the nearby partitions are compared to the query image, search efficiency is significantly improved. Hashing [9, 32, 52, 54, 72–74, 77, 80, 90] learns binary codes for images from global features. Since bitwise operation (XOR) is fast to calculate the Hamming distance, search speed can be boosted greatly.

Many recent works of image search use the convolutional neural network (CNN) [35] to extract image features [1, 2, 17–19, 34, 56, 62, 66, 71]. Instead of representing images with hand-crafted descriptors, these methods encode each image into a unique *global* vector with the designed network. They also provide a high-level description of the content of an image [2]. Compared to local features, they are more suitable for retrieval, because their amount is usually much more smaller for the same collection. For CNN features, hashing [74] seems like a good choice as the indexing structure. However, it still fails to overcome the large-scale issue, since search time still linearly increases with the database volume. When the size of image collection grows to be huge, search time will be unacceptable. Therefore, it motivates us to build efficient indexing for CNN features especially for large-scale image search.

In light of the above discussion, this paper propose an efficient indexing framework for CNN features based on inverted table [76]. Our method draws inspiration from the dictionary building of inverted table [68], which partitions the entire space into (Voronoi) cells via clustering [29]. In this manner, we only need to compare a small number of descriptors in the nearby cells with the query, which helps to reduce much search time. Motivated by this benefit, we reform inverted table for CNN features.

The basic structure of the proposed method is illustrated in Fig. 1. It contains two main procedures. Images are first encoded by a pre-trained CNN into feature vectors. Then, these images are indexed by the proposed inverted indexing framework. The benefits are two-fold. First, a query only needs to be compared with a small number of candidates that locate in the nearby regions, which reduces calculation so as to greatly improve search speed. Second, images are compressedly stored in inverted table, which vastly reduces the storage required.

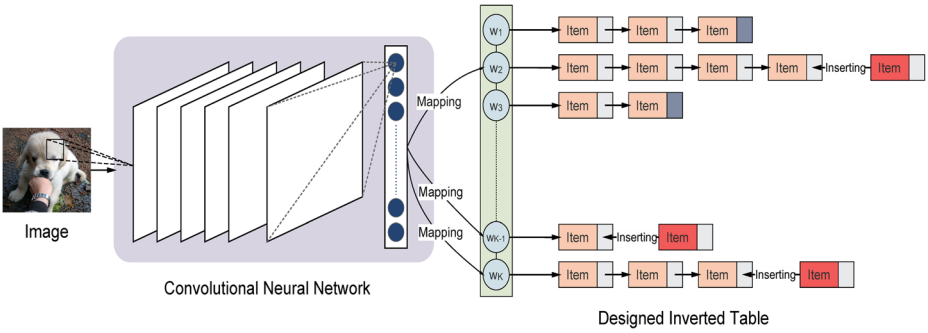


Fig. 1 The basic structure of the proposed method. Each image is encoded by a pre-trained convolutional neural network. Then, the extracted vectors are mapped to multiple codewords and inserted into the corresponding lists. (Best viewed when zoomed in.)

Both the computation cost and storage cost are expected to be reduced for large-scale image search. In summary, our contributions are summarized as follows.

(1) We fully investigate several auxiliary strategies to reform inverted table for CNN features. These strategies are employed for compensating for quantization error, which is the main impediment to effectiveness when an image is represented as a single CNN descriptor. More concretely, embedding codes [27, 76] are calculated as compact representaton of images to prune false matches during search. Besides, multiple assignment (MA) [31] is performed on the query and database images to increase the probability that true matches exist in the candidate set. We demonstrate that both computation cost and storage cost can be reduced with the reformed inverted table. To the best of our knowledge, no similar indexing structure has been published.

(2) We propose a novel indexing framework that combines inverted table and hashing. Inverted table is employed for picking candidates of the query, so that we only calculate the distance between the hash codes of the query and a few candidates. In this manner, search speed is boosted. We demonstrate this framework is faster than the reformed inverted table in large-scale image search.

(3) We fairly compare the performance of the proposed method on several popular CNN features (a short review is provided in Section 2). We demonstrate that the proposed method is widely suitable for various CNN features, and it can significantly improve search speed with a little loss of precision compared to brute-force search.

2 Related work

2.1 The convolutional neural network

The concolutional neural network (CNN) has drawn considerable attention from the computer vision community due to its excellent ability for image classification [23, 35, 67, 69]. All CNNs have similar structure, which is the combination of several convolutional layers and fully connected layers. Krizhevsky *et al.* won the *ILSVRC-2012* competition with an 8-layer CNN named AlexNet [35], which starts the popular research of CNN. Other famous networks include the VGG nets [67], GoogLeNet [69] and ResNet [23]. The depth of the networks is continuously growing. For example, the VGG nets have 16 or 19 layers. For ResNet, its depth has reached 152 layers. Meanwhile, CNN has been widely applied in

many other fields expect for image classification, *e.g.*, object detection [15, 16, 22, 64, 65], image-text matching [25, 87] and person re-identification [85, 88]. In this work, CNN is utilized to extract expressive features for image search.

2.2 Deep learning for retrieval

Feature is the engine of image search. Some previous works [2, 66] have consider using CNN features extracted by the models pre-trained on ImageNet [10, 35] for image retrieval. These works show that CNN features have good ability to represent image content, and search accuracy can be further improved by fine-tuning the model on the relevant datasets. In [86], Zheng et al. conclude that CNN-based method yields competitive accuracy and has advantage in efficiency than SIFT-based method on various retrieval tasks. Motivated by the expressive ability of CNN features, a lot of works [1, 17–19, 34, 42, 43, 56, 62, 71] have been proposed, which utilize deep networks for the retrieval task.

To improve robustness of CNN-based features, several methods aggregate region descriptors to produce compact global features for image retrieval [1, 17–19, 34, 56, 62]. [1] applies sum-pooling to whitened region descriptors. [34] extends [1] by allowing cross-dimensional weighting. Other methods propose hybrid models which involve an encoding technique such as FV [62] and VLAD [17]. Tolias et al. propose R-MAC [71], which produces a global image representation by aggregating the activation features of a CNN in a fixed layout of spatial regions. [18, 19] extend R-MAC by discriminatively learning the representation parameters with the triplet ranking loss and by improving the region pooling mechanism with a region proposal network. Different from the aggregation methods, Li et al. propose a weekly supervised deep matrix factorization scheme [43] to learn hidden representation of images. For image search, although deep learning can be used in another manner such as metric learning [42], feature learning is a main research content of the technique.

Subspace learning [40, 41, 44] projects from the original high-dimensional feature space to a low-dimensional subspace, wherein specific statistical properties can be well preserved. It belongs to non-deep feature learning. CNN-based features are learned straightforward from images, and they provide high-level descriptor of image content [2]. Although CNN features are more compact than local hand-crafted descriptors like SIFT [53], there still exists the issue of large-scale retrieval. Thus, we propose an efficient indexing framework in this paper to handle the issue.

2.3 Inverted table

Inverted table is initially developed to index documents for text retrieval [3]. Sivic et al. [68] utilize inverted table in the vision task that indexes images described by hand-crafted local features via the bog-of-word (BoW) model [59, 60]. In BoW, each image is represented as an orderless collection of visual words. To build an inverted table, a visual dictionary is first constructed via clustering of image features. Then, the ID of an image is inserted into the corresponding lists of the assigned visual words. During search, voting is performed to measure the similarity between images. A complete structure of inverted table can be found in [76].

To efficiently construct a visual dictionary, some fast clustering algorithms are proposed [59, 75]. Partitioned k-means (PKM) [75] splits the entire space into a set of subspaces and then performs a separate k-means clustering process in each subspace. A complete visual dictionary is built by combining different cluster centroids from multiple subspaces. In this manner, PKM can fast build a large visual dictionary using a small training set. A related work of PKM is product quantization (PQ), which estimates distance using

vector-to-centroid distance. PQ decomposes the feature space into a Cartesian product of low-dimensional subspaces and quantizes each subspace separately to reduce computation and storage. [14, 33, 58] are proposed to improve the performance of PQ.

In addition, embedding codes [27, 76] are studied for improving search accuracy of inverted table. These methods generate a short binary code for each descriptor, which is inserted into inverted table with image ID. When searching is performed, embedding codes are used to remove false matches from candidates before voting. In this manner, truly matched images have more chance to rank ahead.

Our method is based on inverted table, i.e., [76]. We draw inspiration from the dictionary building process that partitions the feature space into small regions. Then, a query is expected to be compared with a small number of images in the nearby regions. However, inverted table is more suitable for hand-crafted local features [76]. To solve the problem, we utilize several strategies to reform inverted table for CNN features.

2.4 Hashing

Hashing [9, 72, 73, 77, 80], also known as binary coding, is another kind of indexing technique to solve the large-scale retrieval problem. It compresses global features into equal-length binary codes, and the Hamming distance is used for similarity measure. Search speed on hashing indexing can be very fast, because the Hamming distance can be calculated with bitwise operation (XOR). Besides, it reduces storage space because images are compressively stored as hash codes. Although hashing is quite similar with embedding codes, they are different techniques. The latter is designed for assisting inverted table.

Recent works [5, 12, 36, 37, 39, 45–47, 57, 78, 81–83, 91] utilize deep technique on hash code learning. DH [12] and VDSH [82] exploit a non-linear deep network to produce hash codes for compressing global descriptors. These methods use hand-crafted features, which is similar with traditional hashing [9, 72, 73, 77, 80]. To capture information beneath images, most recent methods are CNN-based hashing models [5, 36, 37, 39, 45–47, 57, 81, 83, 91]. In these works, the input of the hashing network derives from the preceding CNN so as to learn good feature extraction together with hash mapping. For example, Mu et al. propose DeepHash [57], which jointly learns the CNN and hashing network with the proposed exponentiated loss function. Similarly, Li et al. propose DHNNs [39], i.e., a CNN-based hashing approach for large-scale remote sensing image retrieval. DHNNs are composed of a feature network and a hashing network, and the networks can be optimized in the end-to-end training.

Hashing seems a solution to indexing CNN features. However, the linear increase in search time restricts its application and makes it inefficient for searching in a large-scale database. To overcome its shortcoming, we propose a novel indexing framework that combines hashing with inverted table. In this framework, CNN features are encoded as compressed hash codes by hashing and inverted table is used to pick candidates from the gallery. In this manner, only a small number of database images need to be compared with the query by measuring the fast Hamming distance, and search speed is expected to be improved. Note that the framework is designed for hashing that takes hand-crafted features as input. CNN-based hashing is not considered because it generates hash codes directly from input images.

3 Key problem & solution

Although hashing seems a good choice for indexing CNN features, it still suffers from linearly increased search time with the database volume, and it will be inefficient on massive

data. To effectively index large-scale CNN features, we want to build an indexing framework based on inverted table. Some key problems are summarized in the following and we provide our solution after each problem.

Problem 1: How to modify inverted table to make it suitable for CNN features?

In traditional inverted table, images are indexed by storing their IDs in the corresponding lists of visual words according to assignment of their local descriptors. That is, traditional inverted table is tailored for local features. However, CNN features fall into global representation of images, the structure of inverted table needs to be modified to adapt it to CNN features.

Solution. We make simple modification to the original structure of inverted table. For the same set of images, the amount of CNN features is usually much smaller than hand-crafted local features. Therefore, we perform partitioned k-means (PKM) to cluster CNN features into a large visual dictionary. PKM splits a high-dimensional space into several low-dimensional subspaces, and then performs k-means clustering separately in each subspace. Then, it combines visual subdictionaries in different subspaces by the Cartesian product, and finally constructs a complete visual dictionary. In this manner, it can fast build a large visual dictionary using a small number of training descriptors, which is proper for CNN features.

After a visual dictionary is built, the image IDs should be inserted into the inverted table properly. However, each image only contains one CNN feature, the quantization error between CNN features and visual words plays major effect on search accuracy, and it will be more serious than that of hand-crafted local features. For example, dissimilar images may be assigned to the same visual word. Therefore, we need to address the second problem.

Problem 2: How to compensate for the quantization error between CNN features and visual words?

Solution. We utilize several auxiliary strategies to compensate for the quantization error between CNN features and visual words. The ill effect of quantization error is that it reduces the probability of finding the true match, because dissimilar images may be assigned to the same visual word. Similarly, similar images possibly locate in different Voronoi cells so that some true matches will not be picked as candidates. To overcome this problem, we apply two strategies to inverted table.

The first strategy is that we employ multiple assignment (MA) [28] on both database and query images. In this strategy, each CNN feature is assigned to multiple nearest visual words. That is, for an database image, its ID is inserted into multiple lists of the assigned visual words. For an query image, it will look into multiple lists of visual words to pick out candidates. In this manner, the chance of finding out the positive result in the candidate set is increased so as to improve search accuracy.

The second strategy is that we introduce embedding codes [27, 76]. This technique generates a short binary code for each CNN feature based on the assigned visual word, which is inserted into inverted table with the image ID. When searching is performed, embedding codes are used to remove false matches from candidates. In this manner, similar images of the query are more possible to rank ahead.

Problem 3: How to benefit from both inverted table and hashing?

Hashing is tailored for CNN features but suffers from linearly increased search time. Inverted table essentially partitions feature space into small regions, so that only a small number of candidates need to be compared with the query. *Can we combine inverted table*

and hashing so as to benefit from both techniques? Our answer is *yes*, and we propose a new indexing framework.

Solution. We remove the embedding codes from inverted table, and store the hash codes of the database images. During search, the image IDs of the candidates are obtained from inverted table. Then, the hash codes of the candidates are picked out, which are compared with the hash code of the query. In this manner, only a small part of database images are compared with the query so as to reduce much search time.

4 Methodology

4.1 Reform inverted table for CNN features

Suppose there exists an image set $\mathcal{X} = \{x_1, ..., x_N\}$ that contains N images described by D-dim CNN features, where $x_i \in \mathbb{R}^D, i = 1, ..., N$. We want to build efficient indexing for these images with inverted table. To this end, a visual dictionary $\mathcal{W} = \{w_1, ..., w_K\}$ is constructed, where K is the number of visual words. If K is too small, efficiency will be deteriorated since too many images locate in the same Voronoi cell partitioned by $\mathcal{W}$, which will generate a large candidate set. Then, the query image has to be compared with numerous candidates. For this reason, a large visual dictionary is needed. However, the number of CNN features extracted from the same dataset is usually much smaller than hand-crafted local features. It is difficult to build a large visual dictionary. To this end, we perform partitioned k-means (PKM) [75] on CNN features. PKM first splits CNN features into M segments, which usually have equal length, and then performs k-means clustering on each segmented part so as to construct M subdictionaries. Then, the subdictionaries are combined via the Cartesian product to build a large visual dictionary [75], which is next used to construct an inverted table.

Algorithm 1 Building index on IVT_{CNN}.

Input:
Database images $\mathcal{X} = \{x_1, ..., x_N\}$, dictionary size K, MA number S, bit number L.
Output:
Inverted table $ivt_{cnn} = \{\mathcal{W}, \{\mathcal{L}_i\}_{i=1...K}\}$.

```
Construct W = {w_1, ..., w_K} by PKM [75];
for all w_i in W do
    Initialize L_i with an empty list;
end for
for all x_i in X do
    id := the image ID of x_i;
    W* := S nearest visual words of x_i;
    for all w* in W* do
        L* := the corresponding list of w*;
        Calculate the L-bit embedding code e;
        Insert (id, e) into L*;
    end for
end for
```

Consider that each image is represented as a single CNN feature, quantization error will heavily influence search accuracy. For example, similar images may be assigned to different visual words. If each image is only assigned to the nearest visual word, it will have a small chance to find similar images in the corresponding list of the same visual word. To address this problem, we employ multiple assignment (MA) [28], which assigns each CNN feature to multiple visual words. In this manner, each image is inserted into S image lists of inverted table, which will increase the chance of finding the positive result.

Algorithm 2 Performing image search on IVT_{CNN}.

Input:
Query image x_q, $ivt_{cnn} = \{\mathcal{W}, \{\mathcal{L}_i\}_{i=1...K}\}$, MA number W, threshold T.
Output:
Ranked list of the retrieved images $\mathcal{R}$.

```
Construct an empty list ID;
W* := W nearest visual words of x_q in W;
for all w* in W* do
    L* := the corresponding list of w*;
    Calculate the embedding code e_q of x_q;
    for all (id*, e*) in L* do
        if Dist(e_q, e*) ≤ T then
            Insert id* into ID;
        end if
    end for
end for
Count the frequency of IDs in ID;
Rank the retrieved images from high to low frequency;
```

Searching is based on voting. Given a query image x_q, it first looks into the image lists of the W nearest visual words to obtain all the images as candidates. Then, the candidates are sorted by their frequency of occurrence. To further improve search accuracy, embedding codes, such as [27, 76] are also introduced into inverted table. These methods calculate a L-bit short binary code for each CNN feature based on the assigned visual word. During search, in each image list, false matches whose distance to the query is beyond a threshold T are removed. Then, the left images in the W image lists constitute the candidate set. It increases the probability that the positive result ranks ahead.

The structure of the indexing framework that reforms inverted table for CNN features is illustrated in Fig. 2. We denote the framework as IVT_{CNN}. In Algorithm 1 and Algorithm 2, we summarize the process of building index and performing image search on the reformed inverted table, respectively. In these algorithms, the built index is denoted as ivt_{cnn}, which is the collection of a visual dictionary $\mathcal{W}$ and a set of visual-word corresponding lists $\{\mathcal{L}_i\}_{i=1...K}$.

In practice, we find that IVT_{CNN} achieves high precision when the MA numbers S (for database images) and W (for query images) both equal to large values. However, in this situation, search time will be too long. We think that the bottleneck appears in distance calculation of embedding codes. In IVT_{CNN}, the calculation of embedding codes are based on both CNN features and their assigned visual words. The embedding codes of an image in different corresponding lists are different. As a result, a query image has to be repeatedly compared with the images in different lists of the assigned visual words. It is

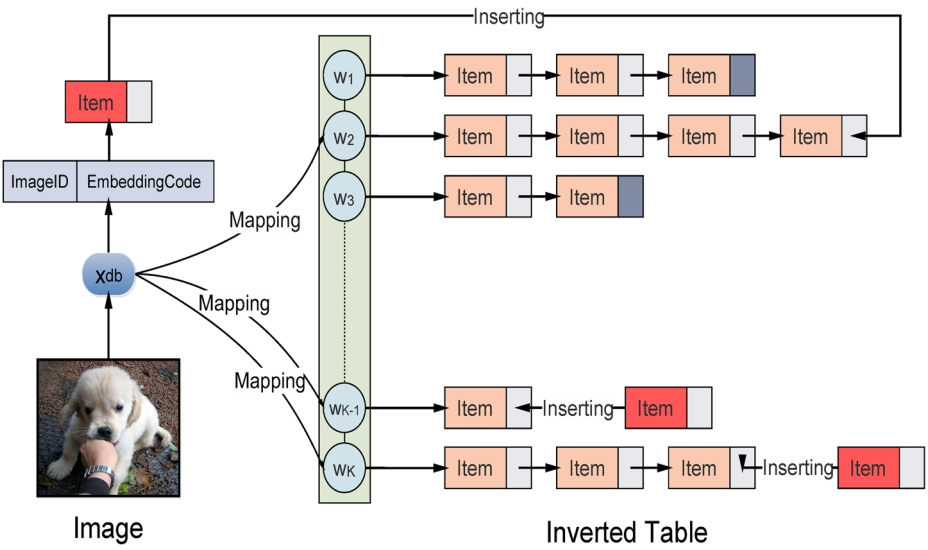


Fig. 2 The structure of IVT_{CNN}, which reforms inverted table for CNN features. Images in the database X_{db} are indexed by inserting their IDs with the embedding codes into the inverted table. Multiple assignment is performed as an auxiliary technique, which assigns each image to multiple visual words so as to improve search accuracy. (Best viewed when zoomed in.)

possible that a query image has been compared with a candidate image for several times. If each image can be represented as a unique binary code, we can only compare each candidate image with the query image only once so as to reduce much search time. For this reason, it motivates us to combine inverted table and hashing to propose a new indexing framework.

Algorithm 3 Building index on $\text{IVT-HASH}_{\text{CNN}}$.

Input:
Database images $\mathcal{X} = \{x_1, ..., x_N\}$, dictionary size K, MA number S, bit number L.
Output:
Inverted table $ivt\text{-}hash_{cnn} = \{\mathcal{W}, \{\mathcal{L}_i\}_{i=1...K}, \mathcal{H}\}$.

```
Construct W = {w_1, ..., w_K} by PKM [75];
for all w_i in W do
    Initialize L_i with an empty list;
end for
Initialize H with an empty set;
for all x_i in X do
    Calculate the L-bit hash code h_i of x_i;
    H = H ∩ {h_i};
    id := the image ID of x_i;
    W* := S nearest visual words of x_i;
    for all w* in W* do
        L* := the corresponding list of w*;
        Insert id into L*;
    end for
end for
```

4.2 Combine inverted table and hashing

We reform inverted table for CNN features to propose IVT_{CNN}. However, we do not modify the search process, so that searching on IVT_{CNN} is still performed by voting. To obtain high search accuracy on IVT_{CNN}, an essential condition should be reached. That is, *the positive result should be assigned to the same visual word with the query as much as possible.* It requires that the MA numbers for both the database and query images (S and W) should equal to large values. Nevertheless, when S and W are large, IVT_{CNN} will lose efficiency.

We think that the use of voting and embedding codes limits the search speed on IVT_{CNN}. Voting requires large values of the MA numbers (S and W) to increase the chance that the positive result and the query are assigned to the same visual word. However, a large S indicates that the image list of each visual word usually contains many images, and a large W indicates that a query looks into many image lists to collect candidates. The enlargement of the two parameters will increase the distance calculation between embedding codes so as to increase search time. Furthermore, the value of embedding codes is also related to visual words [27, 76]. As a result, the embedding codes of the same CNN feature based on different visual words are not identical. For this reason, a query has to be repeatedly compared with the images in different lists of the assigned visual words utilizing the embedding codes. However, it is possible that a query has been compared with a database image for several times. If we can represent each image with a unique binary code, we can compare two images only once so as to reduce much search time.

Algorithm 4 Performing image search on $\text{IVT-HASH}_{\text{CNN}}$

Input:
Query image x_q, $ivt\text{-}hash_{cnn} = \{\mathcal{W}, \{\mathcal{L}_i\}_{i=1...K}, \mathcal{H}\}$, MA number W, threshold T.
Output:
Ranked list of the retrieved images $\mathcal{R}$.

Calculate the hash code h_q of x_q;
Construct an empty **set** $\mathcal{ID}$;
$\mathcal{W}^* := W$ nearest visual words of x_q in $\mathcal{W}$;
for all w^* in $\mathcal{W}^*$ **do**
    $\mathcal{L}^* :=$ the corresponding list of w^*;
    for all id^* in $\mathcal{L}^*$ **do**
        $\mathcal{ID} = \mathcal{ID} \cap \{id^*\}$;
    end for
end for
for all id in $\mathcal{ID}$ **do**
    $h :=$ the hash code of the image whose ID is id;
    if $Dist(h_q, h) \leq T$ **then**
        Remove id from $\mathcal{ID}$;
    end if
end for
Rank the left images in $\mathcal{ID}$ by ascending order of distance to h_q;

In light of the above discussion, we desire to further reform inverted table by discarding voting and storing each image as a unique binary code. To achieve the target, we propose a new indexing framework that combines inverted table and hashing. The benefits of replacing embedding codes with hash codes are two-fold. First, two images are compared only

once. In the proposed framework, candidates are sorted by the distance to the query instead of their frequency of occurrence. In this manner, no matter how many times a database image is found in the candidate set of the query, we need to calculate the distance between their hash codes once. Second, smaller values of the MA number (S and W) are expected. In the new framework, we only need to ensure that the positive result can be found in the candidate set. It is much easier to be reached than the condition that the positive result exists the most frequently in the candidate set. Therefore, we can expect that smaller values of the MA numbers are needed, which will also reduce the computational cost. The structure of the indexing framework that combines inverted table and hashing is illustrated in Fig. 3. The framework is denoted as IVT-HASH$_{\text{CNN}}$. Some minor changes have been made on it compared to IVT$_{\text{CNN}}$. First, we store the hash codes in an independent space out of the inverted table. Second, we only store image IDs in the corresponding lists of the visual words. Both changes help to reduce storage. Third, the retrieved images are ranked according to their Hamming distance to the query. In Algorithm 3 and Algorithm 4, we summarize the algorithms of building index and performing image search on the indexing framework, respectively. The built index is denoted as $ivt\text{-}hash_{cnn}$ in these algorithms. It has three components, i.e., a visual dictionary $\mathcal{W}$, a set of visual-word corresponding lists $\{\mathcal{L}_i\}_{i=1...K}$ and a set of hash codes $\mathcal{H}$.

5 Experiment

We test the proposed method on three instance-retrieval datasets, i.e., Holidays [27], Oxford [63], UKbench [59], and a class-retrieval dataset, i.e., NUS-WIDE [6]. Both small-scale retrieval and large-scale retrieval are tested.

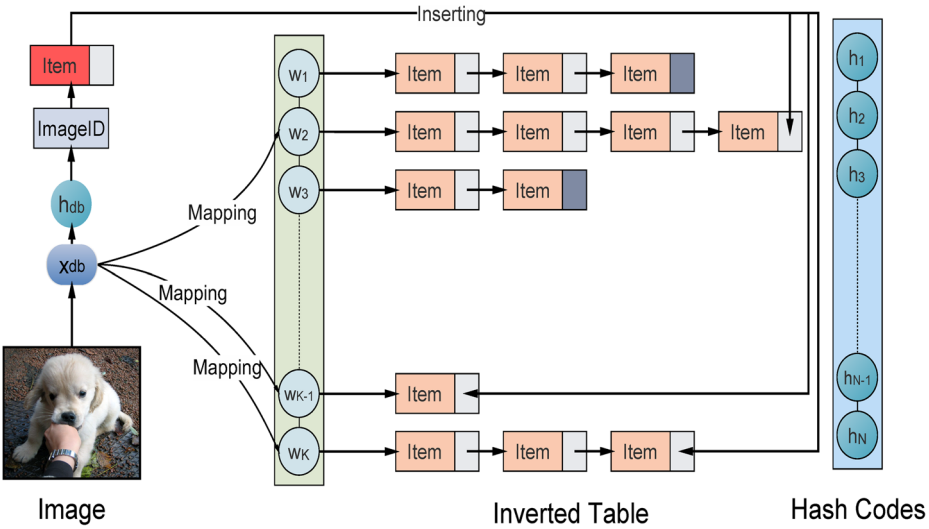


Fig. 3 The structure of IVT-HASH$_{\text{CNN}}$, which combines inverted table and hashing. For each CNN feature x_{db} in the database, we calculate the hash code h_{db} and store it in an hash table. Images are indexed by inserting their image IDs into the inverted table. Multiple assignment is also performed as an auxiliary technique to improve search accuracy. (Best viewed when zoomed in.)

5.1 Datasets

Holiday [27] is an image dataset containing some personal holiday photos. It contains 1,491 images separated into 500 groups. and each group represents a specific scene or object. The first image of each group is the query image and all the other images are the correct retrieval results.

Oxford [63] is a building dataset containing 5,062 images, which are downloaded from Flickr by searching for particular Oxford landmarks. The dataset is manually annotated to generate a comprehensive ground truth for 11 different landmarks, each is given 5 possible queries. Therefore, there are 55 queries in total.

UKbench [59] is a dataset that contains 10,200 images in total. Each 4 images are as a group, which are about the same object from different viewpoints. The first image in each group is as a query, which gives 2,550 queries in total.

NUS-WIDE [6] contains 269,648 images from Flickr, and each image is labeled with at least one of 81 tags. The images of the most frequent 21 tags are picked out to test class retrieval, and the amount is 89,528 in total. These images are partitioned into two subsets, where 5% of the images are used as the query set, and the remaining images are used as the database set. In this dataset, two images are treated as true match if they have at least a common tag.

MIRFlickr [26] is a large-scale dataset that contains 1,000,000 images downloaded from Flickr. We use the dataset in our experiment as distractors to test the performance of large-scale retrieval.

5.2 Experimental setup

Feature: We extract CNN features from popular networks to test the adaptability of the proposed method. The networks include AlexNet [35], VGG nets (VGG16 and VGG19) [67], GoogLeNet [69] and ResNet (ResNet50, ResNet101, ResNet152) [23]. The CNN features of first two networks are extracted from the penultimate fully-connected (FC) layer, which are 4096-dim. The CNN features of the last two networks are extracted from the last FC layer, which are 1000-dim. All the networks are pre-trained on ImageNet [10].

Metric: We use *mean average precision* (MAP) as the search accuracy metric. Given a query and R retrieved results, the value of average precision (AP) is defined as:

$$\mathrm{AP} = \frac{1}{l} \sum_{r=1}^{R} P(r)\iota(r) \tag{1}$$

where l is the number of positive results in the retrieved set. $P(r)$ denotes the precision of the top r retrieved documents, and $\iota(r) = 1$ if the r^{th} retrieved document is a true positive, but $\iota(r) = 0$ otherwise. R is set to the size of database. The values of AP over all queries are averaged to obtain the MAP score. The larger the MAP, the better the accuracy. Note that the query image is removed from the retrieval results to make the MAP value more reasonable.

Search time in the on-line phase is used as the speed metric. It includes the running time to perform the entire search including assignment, voting and ranking. Shorter search time indicates better efficiency. We report the averaged search time to perform a query in our experiment.

Platform: The experiment is performed on a server with an Intel Xeon 2.5GHz CPU, 32GB RAM and Windows Server 2008 R2 x64 operating system. MATLAB R2015a is used as the software to run all the experiment.

5.3 Compared methods

- **Brute Force**. It performs brute-force search using CNN features, and it is the baseline of our method.
- **IVT$_{\mathbf{CNN}}$(HE)**. It is the version of IVT$_{\mathrm{CNN}}$ that uses Hamming embedding (HE) [27] to calculate embedding codes.
- **IVT$_{\mathbf{CNN}}$(LSE)**. It is the version of IVT$_{\mathrm{CNN}}$ that uses linear segment embedding (LSE) [76] to calculate embedding codes.
- **LSH**. Locality sensitive hashing (LSH) [9] maps CNN features into L-dim vectors with a mapping matrix which is generated from the standard normal distribution. Then, it binarizes the vectors into L-bit hash codes.
- **IVT − HASH$_{\mathbf{CNN}}$ (LSH)**. It is the version of IVT-HASH$_{\mathrm{CNN}}$ that uses LSH [9] to learn hash codes.
- **VDSH**. Very Deep Supervised Hashing (VDSH) [82] uses deep neural networks for supervised learning of hash codes.
- **IVT − HASH$_{\mathbf{CNN}}$(VDSH)**. It is the version of IVT-HASH$_{\mathrm{CNN}}$ that uses VDSH [82] to learn hash codes.

The code of VDSH [82] is provided by the authors. The other methods are implemented by ourselves.

5.4 Sensitivity to parameters

We investigate the sensitivity of the proposed method to several important parameters, i.e., the MA number for database images S, the MA number for query images W, the bit number L and the distance threshold T. We perform experiment on three datasets, i.e., Holidays, Oxford and UKbench. IVT$_{\mathrm{CNN}}$(LSE) is tested to observe the sensitivity of IVT$_{\mathrm{CNN}}$ in which we reform inverted table for CNN features. IVT-HASH$_{\mathrm{CNN}}$(LSH) is tested to observe the sensitivity of IVT-HASH$_{\mathrm{CNN}}$ in which we combine inverted table and hashing.

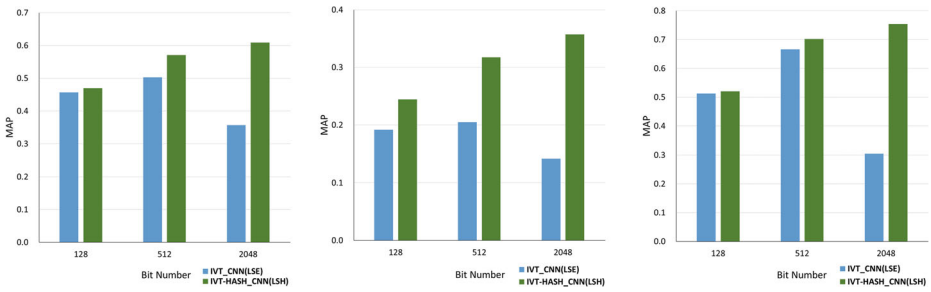
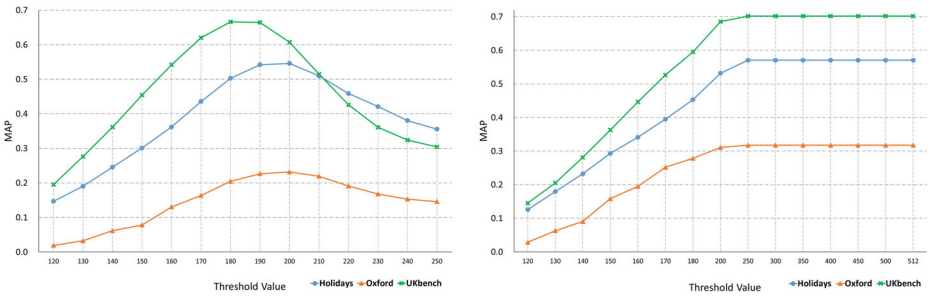


Fig. 4 Variation of MAP of IVT$_{\mathrm{CNN}}$(LSE) and IVT-HASH$_{\mathrm{CNN}}$(LSH) with the bit number L on **a** Holidays, **b** Oxford and **c** UKbench. Images are represented with CNN features extracted from AlexNet. (Best viewed when zoomed in.)

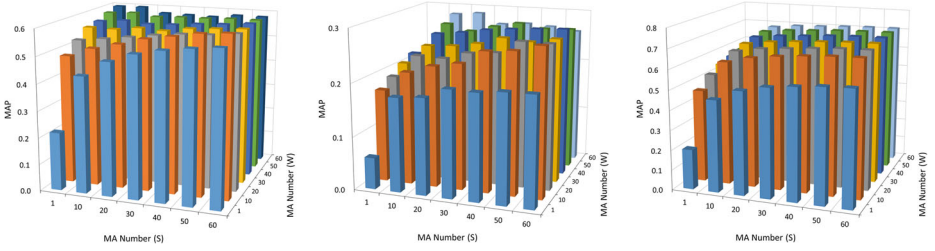


(a) $\mathrm{IVT_{CNN}(LSE)}$ (b) $\mathrm{IVT\text{-}HASH_{CNN}(LSH)}$

Fig. 5 Variation of MAP for **a** $\mathrm{IVT_{CNN}(LSE)}$, **b** $\mathrm{IVT\text{-}HASH_{CNN}(LSH)}$ with the distance threshold T on three datasets, i.e., Holidays, Oxford and UKbench. Images are represented with CNN features extracted from AlexNet. (Best viewed when zoomed in.)

A medium bit number is proper for $\mathrm{IVT_{CNN}(LSE)}$, while a large bit number is suitable for $\mathrm{IVT\text{-}HASH_{CNN}(LSH)}$. As illustrated in Fig. 4, $\mathrm{IVT_{CNN}(LSE)}$ achieves the highest search accuracy on all the three datasets when the bit number L of embedding codes equals to the median value, i.e., 512. It is reasonable since median-length embedding codes help to balance the amount of false candidates and missed candidates. When L is short, dissimilar images may have similar embedding codes so as to collect some false candidates. When L is short, the embedding codes of similar images are possibly very different. Then, some matched images are missed in the candidate set. While for $\mathrm{IVT\text{-}HASH_{CNN}(LSH)}$, the highest search accuracy is achieved when the bit number of hash codes is largest. It fits the conclusion drawn in [9] that a large bit number will improve the probability of mapping similar features into the same hash code. However, a large bit number will also increase the computational cost of distance calculation. To balance search accuracy and speed, L is set to 512 for both the methods in default.

Distance threshold is properly set to the value that equals to about 30% of the bit number for $\mathrm{IVT_{CNN}(LSE)}$. As illustrated in Fig. 5, the best values of the distance threshold T for $\mathrm{IVT_{CNN}(LSE)}$ are 200, 200, 180 on Holidays, Oxford, and UKbench, respectively. These values equal to about 30% of the bit number, i.e., 512. When T is small, the positive result may be removed from candidates by mistake. On the contrary, when T is too large, many dissimilar images are possible reserved in the candidate set, which will reduce search accuracy.



(a) (b) (c)

Fig. 6 Variation of MAP for $\mathrm{IVT_{CNN}(LSE)}$ with the multiple assignment numbers (S and W) on **a** Holidays, **b** Oxford and **c** UKbench. Images are represented with CNN features extracted from AlexNet. (Best viewed when zoomed in.)

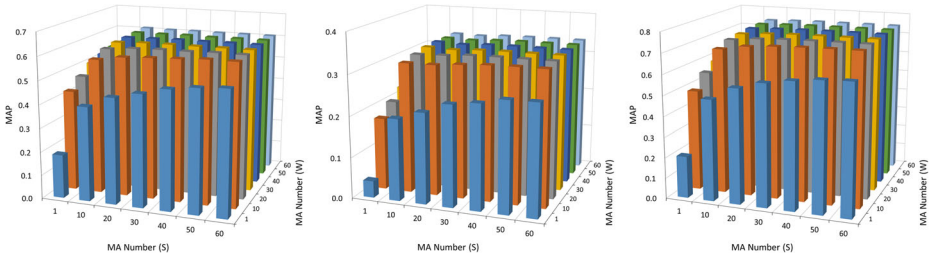


Fig. 7 Variation of MAP for IVT-HASH$_{CNN}$(LSH) with the multiple assignment numbers (S and W) on **a** Holidays, **b** Oxford and **c** UKbench. Images are represented with CNN features extracted from AlexNet. (Best viewed when zoomed in.)

Search accuracy of IVT-HASH$_{CNN}$(LSH) is increased with distance threshold until it equals to a specific value. Then, the performance of IVT-HASH$_{CNN}$(LSH) is stable. As illustrated in Fig. 5, the specific values are 300, 250, 250 on Holidays, Oxford and UKbench, respectively. In IVT-HASH$_{CNN}$(LSH), candidates are ranked according to the Hamming distance of their hash codes to the query. The specific value indicates that the distance between the positive result and query is below it.

Search accuracy is improved for both IVT$_{CNN}$(LSE) and IVT-HASH$_{CNN}$(LSH) when either of multiple assignment (MA) numbers equals to a large value. In Figs. 6 and 7, we show the variation of MAP with different combination values of the MA numbers, i.e., S and W, for IVT$_{CNN}$(LSE) and IVT-HASH$_{CNN}$(LSH), respectively. It can be observed that for both the methods, high search accuracy is achieved when either of the two MA numbers equals to a large value. It is reasonable since enlarging the value of the parameters will increase the probability of finding the positive result in the candidate set. We make S equal to W for for simplicity. The variation of search accuracy and time with equal S and W for IVT$_{CNN}$(LSE) and IVT-HASH$_{CNN}$(LSH) is illustrate in Figs. 8 and 9, respectively. Although search accuracy is improved with increase of the MA numbers, search time is also increased which indicates that both methods lose efficiency. We set the MA numbers to 20 for IVT$_{CNN}$(LSE) and 10 for IVT-HASH$_{CNN}$(LSH), respectively, where we synthetically consider the search accuracy and speed.

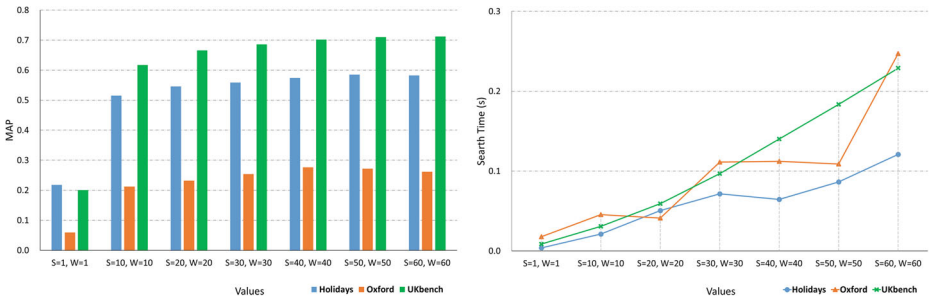


Fig. 8 Variation of **a** MAP and **b** search time for IVT$_{CNN}$(LSE) with the multiple assignment numbers (S and W) when they are equal on Holidays, Oxford and UKbench. Images are represented with CNN features extracted from AlexNet. (Best viewed when zoomed in.)

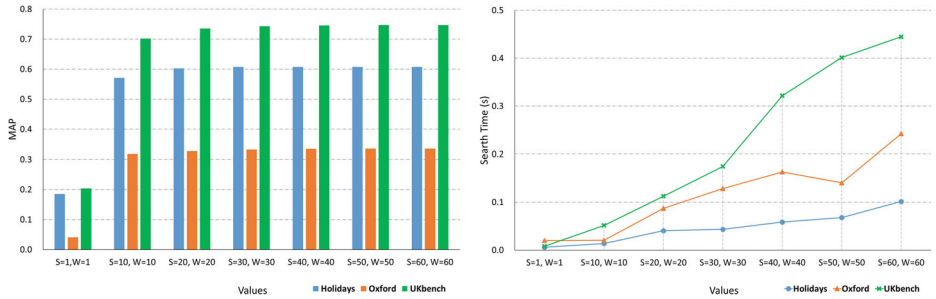


Fig. 9 Variation of **a** MAP and **b** search time for IVT-HASH$_{CNN}$(LSH) with the multiple assignment numbers (S and W) when they are equal on Holidays, Oxford and UKbench. Images are represented with CNN features extracted from AlexNet. (Best viewed when zoomed in.)

5.5 Comparison with baselines on instance-retrieval datasets

We compare the proposed method with several baseline methods on the three instance-retrieval datasets, i.e., Holidays, Oxford and UKbench. Both small-scale and large-scale image search are tested. Besides, we test on different CNN features to validate the adaptability of the methods. The results of search accuracy and time for small-scale and large-scale image search are reported in Tables 1 and 2, respectively. From the results, we arrive at four major conclusions.

(1) IVT-HASH$_{CNN}$(LSH) has similar search accuracy with LSH in most cases. For example, when performing small-scale retrieval on Holidays with CNN features extracted from AlexNet, the MAP score of LSH is 0.6027. Under the same experimental setting, IVT-HASH$_{CNN}$(LSH) achieves 0.5707. Similar result can be observed on large-scale retrieval. Using AlexNet features, LSH achieves the MAP score of 0.4884, and IVT-HASH$_{CNN}$(LSH) achieves 0.4747. The results show that IVT-HASH$_{CNN}$(LSH) can retain the search accuracy of LSH to some extent.

(2) Our method can accelerate search speed especially for large-scale image search. It can be observed in Table 2 that the search time of both IVT$_{CNN}$(LSE) and IVT-HASH$_{CNN}$(LSH) is within a second on all the three datasets no matter which CNN feature is used. These methods are even faster than LSH. Although we use the fast Hamming distance to compare hash codes, the search time of LSH is still linearly increased with the database size. However, in IVT-HASH$_{CNN}$(LSH), only a small part of images are compared with the query, it does not suffer from such a problem.

(3) IVT-HASH$_{CNN}$(LSH) is superior to IVT$_{CNN}$(LSE) in search accuracy and speed. As illustrated in Tables 1 and 2, IVT-HASH$_{CNN}$(LSH) achieves higher search accuracy and smaller search time on all the three datasets, no matter which CNN feature is used to represent images. For example, for small-scale retrieval on UKbench when images are represented as AlexNet features, the MAP score and search time of IVT$_{CNN}$(LSE) are 0.6662 and 0.052s, respectively. While the MAP score and search time of IVT-HASH$_{CNN}$(LSH) are 0.7019 and 0.028s, respectively. Similar observation can be obtained under other experimental settings. Note that the MA numbers for IVT-HASH$_{CNN}$(LSH) are set to 10, which is smaller than that of IVT$_{CNN}$(LSE), i.e., 20. It validates our analysis in Section 4.2 that IVT-HASH$_{CNN}$ requires smaller MA numbers than IVT$_{CNN}$.

Table 1 Search accuracy and time of the baselines and the proposed method on three instance-retrieval datasets, i.e., Holidays, Oxford and UKbench for small-scale image search

Method	Dataset					
	Holidays		Oxford		UKbench	
	MAP	Time (s)	MAP	Time (s)	MAP	Time (s)
AlexNet						
Brute Force	0.5728	0.012	0.3185	0.132	0.7221	0.240
LSH	0.6027	0.012	0.3358	0.034	0.7461	0.049
IVT_{CNN}(HE)	0.4320	0.038	0.1678	0.056	0.5765	0.055
IVT_{CNN}(LSE)	0.5459	0.051	0.2318	0.041	0.6662	0.052
IVT-$HASH_{CNN}$(LSH)	0.5707	0.009	0.3174	0.010	0.7019	0.028
VGG16						
Brute Force	0.6593	0.013	0.2624	0.122	0.8236	0.231
LSH	0.6737	0.011	0.2423	0.030	0.8021	0.052
IVT_{CNN}(HE)	0.5481	0.048	0.1003	0.046	0.3665	0.049
IVT_{CNN}(LSE)	0.5352	0.040	0.1364	0.092	0.6439	0.061
IVT-$HASH_{CNN}$(LSH)	0.6093	0.023	0.2253	0.037	0.7610	0.026
VGG19						
Brute Force	0.6433	0.010	0.2206	0.126	0.8167	0.240
LSH	0.6659	0.034	0.2048	0.052	0.8002	0.053
IVT_{CNN}(HE)	0.5448	0.042	0.0916	0.090	0.3600	0.049
IVT_{CNN}(LSE)	0.5150	0.042	0.1643	0.078	0.6310	0.060
IVT-$HASH_{CNN}$(LSH)	0.5856	0.019	0.1782	0.045	0.7533	0.028
GoogLeNet						
Brute Force	0.6089	0.009	0.2173	0.109	0.7986	0.217
LSH	0.5847	0.074	0.2024	0.076	0.7488	0.049
IVT-$HASH_{CNN}$(LSH)	0.5066	0.017	0.1789	0.027	0.6631	0.019
ResNet50						
Brute Force	0.6836	0.010	0.2677	0.105	0.8664	0.227
LSH	0.6675	0.010	0.2737	0.031	0.8375	0.050
IVT-$HASH_{CNN}$(LSH)	0.5554	0.013	0.2571	0.042	0.7337	0.021
ResNet101						
Brute Force	0.6977	0.010	0.2814	0.106	0.8610	0.222
LSH	0.6729	0.031	0.2148	0.038	0.8422	0.051
IVT-$HASH_{CNN}$(LSH)	0.5725	0.013	0.2070	0.039	0.7312	0.021
ResNet152						
Brute Force	0.6977	0.009	0.2837	0.106	0.8636	0.231
LSH	0.6860	0.008	0.2679	0.027	0.8366	0.050
IVT-$HASH_{CNN}$(LSH)	0.5914	0.023	0.2566	0.031	0.7321	0.018

CNN features extracted from several pre-trained popular networks are compared to validate the adaptability of the methods

Table 2 Search accuracy and time of the baselines and the proposed method on three instance-retrieval datasets, i.e., Holidays, Oxford and UKbench for large-scale image search

Method	Dataset					
	Holidays		Oxford		UKbench	
	MAP	Time (s)	MAP	Time (s)	MAP	Time (s)
AlexNet						
Brute Force	0.4896	210.934	0.2557	241.002	0.6579	231.414
LSH	0.4884	3.324	0.2661	3.486	0.6975	3.449
IVT_{CNN}(HE)	0.2386	0.801	0.0956	0.289	0.5042	1.024
IVT_{CNN}(LSE)	0.3848	0.503	0.1759	0.241	0.6572	0.971
IVT-$HASH_{CNN}$(LSH)	0.4747	0.203	0.2638	0.084	0.6667	0.299
VGG16						
Brute Force	0.4930	229.147	0.2079	258.215	0.7897	233.632
LSH	0.4474	3.285	0.1930	3.429	0.7477	3.314
IVT_{CNN}(HE)	0.1101	0.318	0.0407	0.383	0.2146	0.286
IVT_{CNN}(LSE)	0.1725	0.358	0.0797	0.528	0.5452	0.317
IVT-$HASH_{CNN}$(LSH)	0.4249	0.122	0.1884	0.167	0.7172	0.109
VGG19						
Brute Force	0.4584	240.308	0.1511	257.909	0.7807	242.074
LSH	0.4235	3.424	0.1339	3.301	0.7430	3.314
IVT_{CNN}(HE)	0.1162	0.342	0.0357	0.401	0.2102	0.285
IVT_{CNN}(LSE)	0.1627	0.353	0.0928	0.460	0.5213	0.304
IVT-$HASH_{CNN}$(LSH)	0.3991	0.126	0.1243	0.158	0.7080	0.084
GoogLeNet						
Brute Force	0.3522	246.233	0.1348	265.454	0.7192	239.007
LSH	0.2997	3.273	0.1293	3.315	0.6509	3.298
IVT-$HASH_{CNN}$(LSH)	0.2859	0.085	0.1252	0.108	0.5946	0.057
ResNet50						
Brute Force	0.4725	248.600	0.2161	265.714	0.8324	247.769
LSH	0.4443	3.408	0.1972	3.411	0.7884	3.451
IVT-$HASH_{CNN}$(LSH)	0.4034	0.089	0.1935	0.192	0.7034	0.068
ResNet101						
Brute Force	0.4814	245.295	0.2188	269.339	0.8294	242.765
LSH	0.4591	3.258	0.1522	3.422	0.7980	3.155
IVT-$HASH_{CNN}$(LSH)	0.4211	0.086	0.1515	0.147	0.7044	0.048
ResNet152						
Brute Force	0.4888	248.398	0.2215	267.582	0.8333	250.913
LSH	0.4668	3.266	0.2091	3.422	0.7904	3.448
IVT-$HASH_{CNN}$(LSH)	0.4351	0.082	0.2058	0.182	0.7029	0.062

MIRFlickr is used as distractors. CNN features extracted from several pre-trained popular networks are compared to validate the adaptability of the methods

(4) IVT_{CNN}(LSE) is more sensitive to CNN features than IVT-HASH_{CNN}(LSH). We compare the CNN features extracted from the pre-trained model of several popular networks. As illustrated in Tables 1 and 2, the search-accuracy variation of IVT_{CNN}(LSE) on different features is much greater than IVT-HASH_{CNN}(LSH) in most cases. For example, on Holidays for large-scale retrieval, when CNN features are changed from AlexNet to VGG16, the MAP score of IVT_{CNN}(LSE) drops from 0.3848 to 0.1725. While the MAP score of IVT-HASH_{CNN}(LSH) is changed from 0.4747 to 0.4249, which is changed less. It shows that CNN features have more impact on IVT_{CNN}(LSE).

In summary, our method can improve search speed for large-scale image retrieval. IVT-HASH_{CNN} that combines inverted table and hashing is shown to be superior than IVT_{CNN} that reforms inverted table for CNN features with the two strategies. In Tables 1 and 2, we also compare our method with IVT_{CNN}(HE), i.e., the reformed framework of the inverted-table baseline in [76]. The results validate the superiority of IVT-HASH_{CNN}, since it has fast search speed with a little loss of search accuracy. Note that another benefit of the proposed method is that it can reduce storage cost, because it only stores subdictionaries, image IDs and binary codes.

However, there is limitation of the proposed method. For IVT_{CNN}, it requests the feature dimension to be an integral multiple of the bit number of embedding codes such as [27, 76]. For this reason, we do not report the performance of reformed inverted tables when CNN features are extracted from GoogLeNet and ResNet. Besides, this method is not robust to CNN features. For IVT-HASH_{CNN}, its search accuracy is probably limited by the hashing method that is used to calculate hash codes. And only the hashing methods that take hand-crafted features as input are proper for this method.

In Fig. 10, we illustrate some visual retrieval results of our method, i.e., IVT-HASH_{CNN}(LSH), on Holidays. The results indicate that our method outputs reasonable rank lists.

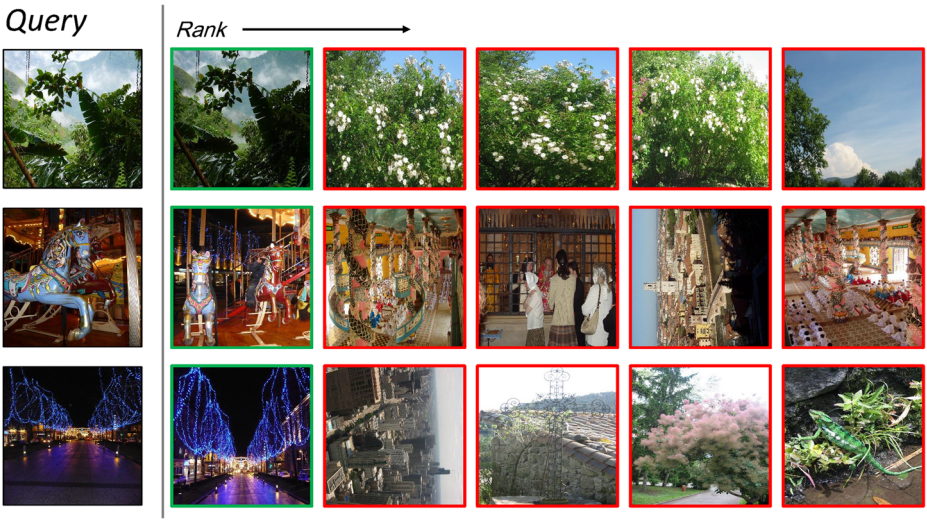


Fig. 10 Retrieval results of the proposed method on Holidays. The results are sorted from left to right according to their similarity to the query (high to low). The images in green boxes are true matches, and the images in red boxes are false matches. Images are represented as CNN features extracted from AlexNet. We observe that our method outputs reasonable rank lists

5.6 Comparison with baseline on a class-retrieval dataset

We have observed that our method, i.e., IVT-HASH$_{CNN}$, can build efficient indexing of CNN features for large-scale instance-level image retrieval when it uses LSH [9] to calculate hash codes. In this section, we test whether the method is efficient for large-scale class-level image retrieval. To make IVT-HASH$_{CNN}$ work for class-level image retrieval, we use VDSH [82] to calculate hash codes. VDSH is a state-of-the-art supervised hashing method that uses deep networks to learn hash functions for hand-crafted features, which is suitable to be utilized in IVT-HASH$_{CNN}$. The new method is denoted as IVT-HASH$_{CNN}$(VDSH). We compare it with the baseline hashing method, i.e., VDSH on a class-retrieval dataset, i.e., NUS-WIDE [6]. We report the search accuracy and time in Table 3.

Table 3 Search accuracy and time of the baseline and the proposed method on a class-retrieval dataset, i.e., NUS-WIDE for both small-scale and large-scale image search

Method	Dataset			
	NUS-WIDE		NUS-WIDE + MIRFlickr	
	mAP	Time (s)	mAP	Time (s)
AlexNet				
VDSH	0.5862	0.254	0.4032	1.675
IVT-HASH$_{CNN}$(VDSH)	0.5865	0.043	0.3812	0.096
VGG16				
VDSH	0.5893	0.305	0.3393	2.167
IVT-HASH$_{CNN}$(VDSH)	0.5915	0.050	0.3204	0.103
VGG19				
VDSH	0.5908	0.315	0.3615	0.919
IVT-HASH$_{CNN}$(VDSH)	0.5911	0.059	0.3341	0.106
GoogLeNet				
VDSH	0.5031	0.313	0.3144	1.826
IVT-HASH$_{CNN}$(VDSH)	0.5057	0.037	0.2912	0.090
ResNet50				
VDSH	0.5639	0.315	0.3378	2.300
IVT-HASH$_{CNN}$(VDSH)	0.5652	0.047	0.3159	0.098
ResNet101				
VDSH	0.5621	0.318	0.3515	2.096
IVT-HASH$_{CNN}$(VDSH)	0.5643	0.069	0.3308	0.105
ResNet152				
VDSH	0.5637	0.315	0.3606	2.224
IVT-HASH$_{CNN}$(VDSH)	0.5646	0.048	0.3421	0.098

MIRFlickr is used as distractors for large-scale image search. CNN features extracted from several pre-trained popular networks are compared to validate the adaptability of the methods

Minor variation of the search accuracy metric A major difference between NUS-WIDE and the three instance-retrieval datasets, i.e., Holidays, Oxford and UKbench is that it has a large number of matched images for each query in the ground truth. However, our method only returns a small number of candidate images, since it is based on inverted table. If we still use MAP as the search accuracy metric to evaluate on all the groundtruch images, it will be unfair for our method because it only returns a short list of top-position retrieved images. In order to solve the problem, we use mAP instead of MAP to measure search accuracy. mAP is short for mean average precision, whose AP score is still calculated as in (1). The difference is that in MAP, R equals to the database size, but in mAP, R equals to an established value. mAP evaluates the search accuracy of top-R retrieved results, which is widely used in cross-media hashing such as [48, 49]. In our experiment, we set R to 50.

IVT-HASH$_{\text{CNN}}$(VDSH) is efficient to index CNN features for large-scale class-level image retrieval. We observe that IVT-HASH$_{\text{CNN}}$(VDSH) achieves lower search accuracy but is faster than VDSH when we perform large-scale retrieval. For example, as illustrated in Table 3, when images are represented with the CNN features extracted from AlexNet, the mAP score of VDSH is 0.4032 and the search time is 1.675s for large-scale retrieval. Under the same setting, the mAP score of IVT-HASH$_{\text{CNN}}$(VDSH) is 0.3812 and the search time is 0.096s. Similar observation can be obtained when images are represented with the CNN features extracted from the other networks. The results indicate that IVT-HASH$_{\text{CNN}}$ is suitable for large-scale class-level image retrieval as well.

6 Conclusion

In this paper, we propose an efficient indexing framework to build index for large-scale CNN features, and it is based on inverted table. Our contributions are two-fold. First, we investigate two strategies to reform inverted table for CNN features. Second, we propose a new framework that benefits from both inverted table and hashing. Experiment on four datasets, i.e., Holidays, Oxford, UKbench and NUS-WIDE, demonstrates that our method is efficient for large-scale image search. The limitation of the proposed method is that it can only utilize hashing designed for hand-crafted features to help building index for CNN features. However, most of the recent hashing methods extract hash codes directly from images with CNNs [5, 36, 37, 39, 45–47, 57, 81, 83, 91]. How to accelerate such hashing methods with a similar structure to our method is the future study of ours. Besides, as another future work, we plan to introduce the proposed method into cross-media retrieval [48, 49] to build efficient index for large-scale image-text embeddings.

Acknowledgments This work was supported in part by National Natural Science Foundation of China (No.61532005, No.61572065), National Key Research and Development of China (No.2016YFB08004 04, 2017YFC1703503), Joint Fund of Ministry of Education of China and China Mobile (No.MCM20160102).

References

1. Babenko A, Lempitsky V (2015) Aggregating local deep features for image retrieval. In: ICCV, pp 1269–1277
2. Babenko A, Slesarev A, Chigorin A, Lempitsky V (2014) Neural codes for image retrieval. In: ECCV, pp 584–599
3. Baeza-Yates R, Ribeiro-Neto B et al (1999) Modern information retrieval, vol 463
4. Bay H, Tuytelaars T, Van Gool L (2006) Surf: Speeded up robust features, pp 404–417

5. Cao Z, Long M, Wang J, Yu PS (2017) Hashnet: Deep learning to hash by continuation. arXiv:1702.00758
6. Chua TS, Tang J, Hong R, Li H, Luo Z, Zheng Y (2009) Nus-wide: a real-world web image database from national university of singapore. In: CIVR
7. Ciaccia P, Patella M, Zezula P (1997) M-tree: An efficient access method for similarity search in metric spaces. In: VLDB
8. Dalal N, Triggs B (2005) Histograms of oriented gradients for human detection. In: CVPR, vol. 1, pp 886–893
9. Datar M, Immorlica N, Indyk P, Mirrokni VS (2004) Locality-sensitive hashing scheme based on p-stable distributions. In: SoCG, pp 253–262
10. Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009) Imagenet: A large-scale hierarchical image database. In: CVPR, pp 248–255
11. Dong L, Liang Y, Kong G, Zhang Q, Cao X, Izquierdo E (2016) Holons visual representation for image retrieval. TMM 18(4):714–725
12. Erin Liong V, Lu J, Wang G, Moulin P, Zhou J (2015) Deep hashing for compact binary codes learning. In: CVPR, pp 2475–2483
13. Gao Z, Xue J, Zhou W, Pang S, Tian Q (2016) Democratic diffusion aggregation for image retrieval. TMM 18(8):1661–1674
14. Ge T, He K, Ke Q, Sun J (2014) Optimized product quantization. TPAMI 36(4):744–755
15. Girshick R (2015) Fast r-cnn. In: ICCV, pp 1440–1448
16. Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR, pp 580–587
17. Gong Y, Wang L, Guo R, Lazebnik S (2014) Multi-scale orderless pooling of deep convolutional activation features. In: ECCV, pp 392–407
18. Gordo A, Almazán J., Revaud J, Larlus D (2016) Deep image retrieval: Learning global representations for image search. In: ECCV, pp 241–257
19. Gordo A, Almazan J, Revaud J, Larlus D (2017) End-to-end learning of deep visual representations for image retrieval. IJCV 124(2):237–254
20. Guttman A (1984) R-trees: A dynamic index structure for spatial searching, vol 14
21. Han J, Ma KK (2002) Fuzzy color histogram and its use in color image retrieval. TIP 11(8):944–952
22. He K, Zhang X, Ren S, Sun J (2015) Spatial pyramid pooling in deep convolutional networks for visual recognition. TPAMI 37(9):1904–1916
23. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: CVPR, pp 770–778
24. Hu MK (1962) Visual pattern recognition by moment invariants. IRE Trans Inf Theory 8(2):179–187
25. Hu Y, Zheng L, Yang Y, Huang Y (2017) Twitter100k: A real-world dataset for weakly supervised cross-media retrieval. TMM
26. Huiskes MJ, Lew MS (2008) The mir flickr retrieval evaluation. In: MIR, pp 39–43
27. Jegou H, Douze M, Schmid C (2008) Hamming embedding and weak geometric consistency for large scale image search. In: ECCV, pp 304–317
28. Jégou H., Douze M, Schmid C (2010) Improving bag-of-features for large scale image search. IJCV 87(3):316–336
29. Jegou H, Douze M, Schmid C (2011) Product quantization for nearest neighbor search. TPAMI 33(1):117–128
30. Jégou H, Douze M, Schmid C, Pérez P (2010) Aggregating local descriptors into a compact image representation. In: CVPR, pp 3304–3311
31. Jegou H, Harzallah H, Schmid C (2007) A contextual dissimilarity measure for accurate and efficient image search. In: CVPR, pp 1–8
32. Ji J, Li J, Yan S, Zhang B, Tian Q (2012) Super-bit locality-sensitive hashing. In: NIPS, pp 108–116
33. Kalantidis Y, Avrithis Y (2014) Locally optimized product quantization for approximate nearest neighbor search. In: CVPR, pp 2329–2336
34. Kalantidis Y, Mellina C, Osindero S (2016) Cross-dimensional weighting for aggregated deep convolutional features. In: ECCV, pp 685–701
35. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: NIPS, pp 1097–1105
36. Lai H, Pan Y, Liu Y, Yan S (2015) Simultaneous feature learning and hash coding with deep neural networks. In: CVPR, pp 3270–3278
37. Li Q, Sun Z, He R, Tan T (2017) Deep supervised discrete hashing. In: NIPS, pp 2479–2488
38. Li Y, Wang S, Tian Q, Ding X (2015) A survey of recent advances in visual feature detection. Neurocomputing 149:736–751

39. Li Y, Zhang Y, Huang X, Zhu H, Ma J (2018) Large-scale remote sensing image retrieval by deep hashing neural networks. TGRS 56(2):950–965
40. Li Z, Liu J, Tang J, Lu H (2015) Robust structured subspace learning for data representation. TPAMI 37(10):2085–2098
41. Li Z, Tang J (2015) Unsupervised feature selection via nonnegative spectral analysis and redundancy control. TIP 24(12):5343–5355
42. Li Z, Tang J (2015) Weakly supervised deep metric learning for community-contributed image retrieval. TMM 17(11):1989–1999
43. Li Z, Tang J (2017) Weakly supervised deep matrix factorization for social image understanding. TIP 26(1):276–288
44. Li Z, Tang J, He X (2017) Robust structured nonnegative matrix factorization for image representation. TNNLS
45. Lin K, Lu J, Chen C, Zhou J (2016) Learning compact binary descriptors with unsupervised deep neural networks, pp 1183–1192
46. Lin K, Yang HF, Hsiao JH, Chen C (2015) Deep learning of binary hash codes for fast image retrieval. In: CVPR, pp 27–35
47. Liu H, Wang R, Shan S, Chen X (2016) Deep supervised hashing for fast image retrieval, pp 2064–2072
48. Liu R, Wei S, Zhao Y, Zhu Z, Wang J (2018) Multi-view cross-media hashing with semantic consistency. IEEE MultiMedia
49. Liu R, Zhao Y, Wei S, Zhu Z (2015) Cross-media hashing with centroid approaching. In: ICME, pp 1–6
50. Liu Y, Zhang D, Lu G, Ma WY (2007) A survey of content-based image retrieval with high-level semantics. Pattern Recogn 40(1):262–282
51. Liu Z, Li H, Zhou W, Hong R, Tian Q (2015) Uniting keypoints: Local visual information fusion for large-scale image search. TMM 17(4):538–548
52. Liu Z, Li H, Zhou W, Zhao R, Tian Q (2014) Contextual hashing for large-scale image search. TIP 23(4):1606–1614
53. Lowe DG (2004) Distinctive image features from scale-invariant keypoints. IJCV 60(2):91–110
54. Lv Y, Ng WW, Zeng Z, Yeung DS, Chan PP (2015) Asymmetric cyclical hashing for large scale image retrieval. TMM 17(8):1225–1235
55. Mikolajczyk K, Schmid C (2005) A performance evaluation of local descriptors. TPAMI 27(10):1615–1630
56. Mohedano E, McGuinness K, O'Connor NE, Salvador A, Marqués F, Giró-i Nieto X (2016) Bags of local convolutional features for scalable instance search. In: ACMMM, pp 327–331
57. Mu Y, Liu Z (2017) Deep hashing: A joint approach for image signature learning. In: AAAI, pp 2380–2386
58. Ning Q, Zhu J, Zhong Z, Hoi SC, Chen C (2017) Scalable image retrieval by sparse product quantization. TMM 19(3):586–597
59. Nister D, Stewenius H (2006) Scalable recognition with a vocabulary tree. In: CVPR, vol 2, pp 2161–2168
60. Nowak E, Jurie F, Triggs B (2006) Sampling strategies for bag-of-features image classification. In: ECCV, pp 490–503
61. Perronnin F, Dance C (2007) Fisher kernels on visual vocabularies for image categorization. In: CVPR, pp 1–8
62. Perronnin F, Larlus D (2015) Fisher vectors meet neural networks: A hybrid classification architecture. In: CVPR, pp 3743–3752
63. Philbin J, Chum O, Isard M, Sivic J, Zisserman A (2007) Object retrieval with large vocabularies and fast spatial matching. In: CVPR, pp 1–8
64. Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: Unified, real-time object detection. In: CVPR, pp 779–788
65. Ren S, He K, Girshick R, Sun J (2015) Faster r-cnn: Towards real-time object detection with region proposal networks. In: NIPS, pp 91–99
66. Sharif Razavian A, Azizpour H, Sullivan J, Carlsson S (2014) Cnn features off-the-shelf: an astounding baseline for recognition. In: CVPR, pp 512–519
67. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556
68. Sivic J, Zisserman A (2003) Video google: A text retrieval approach to object matching in videos. In: ICCV, pp 1470–1477
69. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: CVPR, pp 1–9

70. Tamura H, Mori S, Yamawaki T (1978) Textural features corresponding to visual perception. IEEE Trans Syst Man Cybern 8(6):460–473
71. Tolias G, Sicre R, Jégou H (2015) Particular object retrieval with integral max-pooling of cnn activations. arXiv:1511.05879
72. Wang J, Kumar S, Chang SF (2012) Semi-supervised hashing for large-scale search. TPAMI 34(12):2393–2406
73. Wang J, Wang J, Yu N, Li S (2013) Order preserving hashing for approximate nearest neighbor search. In: ACMMM, pp 133–142
74. Wang J, Zhang T, Sebe N, Shen HT, et al. (2017) A survey on learning to hash. TPAMI
75. Wei S, Wu X, Xu D (2013) Partitioned k-means clustering for fast construction of unbiased visual vocabulary. In: The Era of Interactive Media, pp 483–493
76. Wei S, Xu D, Li X, Zhao Y (2013) Joint optimization toward effective and efficient image search. IEEE Trans Cybern 43(6):2216–2227
77. Weiss Y, Torralba A, Fergus R (2009) Spectral hashing. In: NIPS, pp 1753–1760
78. Xia R, Pan Y, Lai H, Liu C, Yan S (2014) Supervised hashing for image retrieval via image representation learning. In: AAAI, vol. 1, p. 2
79. Yang Y, Nie F, Xu D, Luo J, Zhuang Y, Pan Y (2012) A multimedia retrieval framework based on semi-supervised ranking and relevance feedback. TPAMI 34(4):723–742
80. Zhang D, Wang J, Cai D, Lu J (2010) Self-taught hashing for fast similarity search. In: SIGIR, pp 18–25
81. Zhang J, Peng Y, Zhang J (2016) Ssdh: Semi-supervised deep hashing for large scale image retrieval. arXiv:1607.08477
82. Zhang Z, Chen Y, Saligrama V (2016) Efficient training of very deep neural networks for supervised hashing. In: CVPR, pp 1487–1495
83. Zhao F, Huang Y, Wang L, Tan T (2015) Deep semantic ranking based hashing for multi-label image retrieval. In: CVPR, pp 1556–1564
84. Zheng L, Wang S, Liu Z, Tian Q (2015) Fast image retrieval: query pruning and early termination. TMM 17(5):648–659
85. Zheng L, Yang Y, Hauptmann AG (2016) Person re-identification: Past, present and future. arXiv:1610.02984
86. Zheng L, Yang Y, Tian Q (2017) Sift meets cnn: A decade survey of instance retrieval. TPAMI
87. Zheng Z, Zheng L, Garrett M, Yang Y, Shen YD (2017) Dual-path convolutional image-text embedding. arXiv:1711.05535
88. Zheng Z, Zheng L, Yang Y (2017) A discriminatively learned cnn embedding for person reidentification. ToMM 14(1):13
89. Zhong Z, Zhu J, Hoi SC (2015) Fast object retrieval using direct spatial matching. TMM 17(8):1391–1397
90. Zhou W, Yang M, Li H, Wang X, Lin Y, Tian Q (2014) Towards codebook-free: Scalable cascaded hashing for mobile image search. TMM 16(3):601–611
91. Zhuang B, Lin G, Shen C, Reid I (2016) Fast training of triplet-based deep binary embedding networks. In: CVPR, pp 5955–5964

Ruoyu Liu is a PhD student of the Institute of Information Science, Beijing Jiaotong University. His research interests include multimedia retrieval, computer vision and deep learning.

Shikui Wei is currently a Professor with the Institute of Information Science, Beijing Jiaotong University. His research interests include computer vision, image/video analysis and retrieval, and copy detection.

Yao Zhao is now the Director of the Institute of Information Science, Beijing Jiaotong University. His research interests include image/video coding, digital watermarking and forensics, and video analysis and understanding.

Yi Yang is now a professor with the Faculty of Engineering and Information Technology, University of Technology Sydney (UTS). His current research interest includes machine learning and its applications to multimedia content analysis and computer vision.