Large Scale Near-duplicate Image Retrieval via Patch Embedding

Shangpeng Yan, Xiaoyun Zhang, Wenbo Bao, Li Chen, Zhiyong Gao
Institute of Image Communication and Network Engineering, Shanghai Jiao Tong University
{ysp123, xiaoyun.zhang, baowenbo, hilichen, zhiyong.gao}@sjtu.edu.cn

Abstract

Large scale near-duplicate image retrieval (NDIR) relies on the Bag-of-Words methodology which quantizes local features into visual words. However the direct match of these visual words typically leads to unpleasant mismatches due to quantization errors. To enhance the discriminability of the matching process, existing methods usually exploit hand-crafted contextual information, which have limited performance in complicated real-world scenarios. In contrast, we in this paper propose a trainable lightweight embedding network to extract local binary features. The network takes image patches as inputs and generates the binary code that can be efficiently stored in the inverted indexing file and helps discard mismatches immediately during the retrieval process. We improve the discriminability of the code by elaborately composing the training patches for network optimization, which consists of a proper inter-class (non-duplicate) patches selection and a rich intra-class (near-duplicate) patches generation. We evaluate our approach on the open NDIR dataset, INRIA CopyDays, and the experimental results show that our method performs favorably against the state-of-the-art algorithms. Furthermore, with a relatively short code length, our approach achieves higher query speed and lower storage occupation.

Figure 1: Sample patches which are quantized into the same visual word. These patches are similar in structure while we need to distinguish them in the view of NDIR.

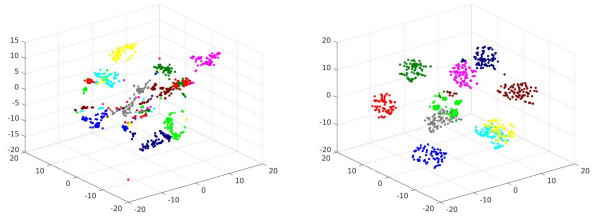

Figure 2: The T-sne[15] distribution of sift features (left) and our learned features (right) for the samples in Figure 1. Each color represents a different patch. The same color represents the patch and its distortions. It can be seen that our learned features are more discriminative.

1. Introduction

Near-duplicate image retrieval (NDIR) problem aims to find the distorted versions in the large database for the giving image. Distortions performed on the original image include cropping, rotation, scaling, compression, watermarking and text adding. As a sub-task of the image retrieval, it's useful in some practical applications such as copyright infringement detection and keyframe identification. Due to the particularity of this problem, practical applications of NDIR usually have strict demand on the performance such as high retrieval accuracy or true positive rate.

Traditional methods tackling the image retrieval are based on the Bag-of-Words model[20]. In the Bag-of-Words model, local descriptors such as SIFT[12] are firstly quantized into visual words, which builds a codebook for retrieval. And then according to the size of the codebook, various algorithms are proposed to encode a query image. For the small size codebook, usually a global descriptor is constructed such as VLAD[6] and Fisher vector[16]. For the large scale image retrieval problem, which this paper mainly focuses on, the codebook is huge with millions of visual words, thus clustering-based quantization and inverted index is generally applied to enable efficient storage and retrieval. However, because of the quantization error, the retrieval accuracy of direct match using visual words will decrease. And to improve retrieval performance, different methods are designed for post-verification

and mismatch removing. Geometric verification is the most popular technique[17][14][25][23][21], where mismatches are filtered out based on the first-round retrieval result. Some works [22][11][24] adopt additional feature descriptors, which are usually hand-crafted, into visual words, and remove mismatches immediately during image retrieval.

As for near-duplicate image retrieval, the quantization error effect becomes worse since similar but not near-copy image patches may be clustered as the same visual word and lead to more mismatches. As shown in Fig. 1, SIFT descriptors of 10 similar patches with their corresponding near-copy distorted versions are quantized into the same cluster and thus are represented by the same visual word, which is not appropriate for NDIR. These 10 patches are just similar and should not be regarded as near-duplicate versions. Instead, these 10 patches are different and should stay apart in the feature space as 10 clusters, where each cluster corresponds to one patch and its distorted versions, as shown in Fig. 2.

Inspired by the Convolutional Neural Networks (CNN), we adopt the learning based descriptors as the additional feature for visual words to improve the discriminability for NDIR. As shown in Fig. 3, the proposed embedding network takes various of distorted patches as input training samples. Each selected patch and its distorted versions are regarded as one class and should stay close in the embedding space. Any two different patches and their corresponding distortions should stay far away, since they belong to two classes within the context of near-duplicate retrieval. Specifically, discriminative learning based on triplet patches and hash learning for binarization are alternatively trained to output a compact binary descriptor for NDIR. And another advantage of our proposed method is that patch samples for training can be easily generated from a small set of raw images.

We summarize the main contributions of this paper as follows:

1. We propose a patch-based embedding framework for the NDIR task.

2. We combine the discriminative learning and hash learning to encode the image patch to the local binary feature suitable for large scale NDIR.

3. We conduct extensive experiments and demonstrate that the proposed method performs favorably against the state-of-the-art algorithms.

2. Related work

Large scale image retrieval has been well developed in the past few years. The induction of local discrminative feature such as SIFT[12] and the application of Bag-of-Words

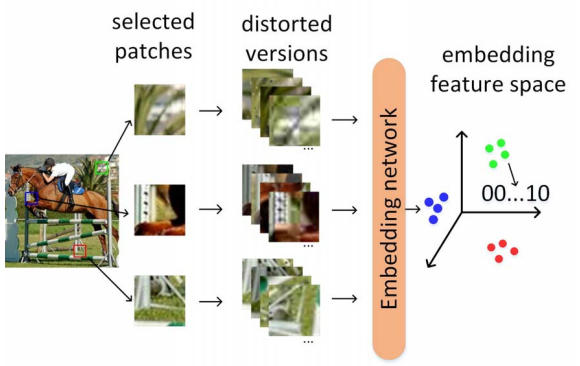

Figure 3: A brief view of our proposed method. We generate various distorted versions to simulate the possible transform in real scenario. After encoding the patch features are separated in the embedding space. The binary features help fast remove mismatches with Hamming distance threshold.

model greatly improve the speed and accuracy of image retrieval. For the large scale NDIR task, visual words based local descriptors precise match is the core of most existing methods. In this section we give a brief review of current methods in improving the visual words match performance.

Full geometric verification such as RANSAC [17] is a basic method to improve the retrieval precision. It calculates the affine transformations for random selected points repeatedly and remove the outliers. RANSAC is effective for re-ranking candidate images but has efficiency problems. To improve retrieval speed a spatial-coding method[25] is proposed to use the relative positions of local descriptors. By a specially designed relative position map, the verification can be achieved with logical X-OR operation and the speed is largely increased. In LMR[14] a novel hand crafted match representation is constructed. It majors in removing mismatches between two images and treat the task as a two-class classification.

Some works remove mismatches immediately during retrieval process by inserting additional feature into visual words. Liu [11] designs a local contextual binary feature for visual words and combine it with the inverted index. It builds a relationship dictionary and quantizes the relationship between two features with the relationship dictionary. Chen [1] propose a novel NDIR framework based on visual phrase. A spatial visual phrase (SVP) model is introduced to utilize the geometric information among visual words. Yao [22] promotes the contextual feature and propose a more stable angle encoding based descriptor. It carefully selects the contextual points with respect to the reference feature and acquire the contextual feature by encoding the relative angles between these points.

In our approach we adopt the methodology of immediate mismatch removal by inserting additional feature into

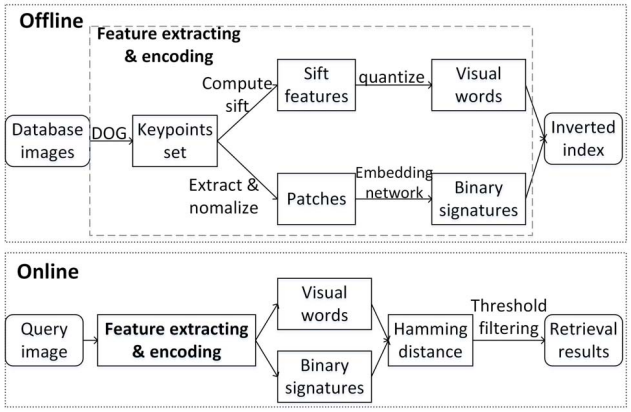

Figure 4: Near duplicate image retrieval framework. In offline section we extract and store features from database images. In online section the retrieval results are returned for the query image.

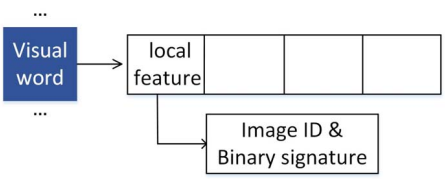

Figure 5: The inverted index structure.

visual words and use learned binary descriptor as the additional feature. We show that out approach is precise in visual words match and low in the computation cost.

3. NDIR framework

In this section we present an overall description of our approach for the NDIR task. Fig. 4 shows the retrieval framework. The whole processing includes offline and online two sections. We will first explain the two sections and then we give a detailed description of how we make the training patches for the embedding network.

3.1. Offline section

Our approach is based on the BoW model and inverted index. To build the inverted index we need first to train a codebook by clustering sift features. Each visual word in the codebook lies in the center of a cluster. Due to the large codebook size, the direct k-means clustering is not appropriate. In our work we choose approximate k-means (AKM) [17] method to build the codebook.

For a given raw image, we use DOG[12] detector to extract a set of keypoints. We compute sift features and the learned binary features at the same time. To obtain the binary features we need first extract the image patches and feed patches into the network. The region of patch is significant since we expect that same regions should be extracted from the same location of two near-duplicate images. Considering the scaling case in NDIR, the regions are supposed to be dynamic and can be self-adjusted. In our approach we utilize the DOG extreme point scaling information to determine the selected region. This help us to unify the patch size and guarantees we select the near-duplicate patches from near-duplicate images. Finally the patches are normalized and reshaped into the network input shape. We feed the normalized patches into network to obtain the binary signatures.

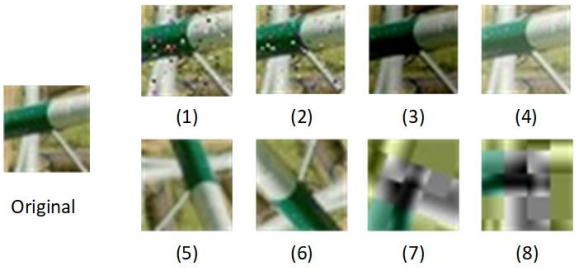

Figure 6: Examples of distortion we adopt during training. (1,2) add noise. (3,4) illuminance changes. (5,6) rotate. (7,8) JPEG compression. The scope may change in the process such as the (6) contains only part of the original patch. We use combinations of these processings to boost the variety of training set.

The inverted index structure is shown in Fig. 5 . For each keypoint in an image we quantize the corresponding sift feature to get the visual word and store the image ID and the corresponding binary signature in the word entry.

3.2. Online section

For a query image we extract and encode features in the same way of offline section. And for each quantized sift feature in the query image we traverse the inverted list corresponding to its assigned visual word. We calculate the Hamming distance of the query binary signature with each signature in the inverted list and verify if it is a near-duplicate match with a threshold. Lower threshold leads to lower recall and higher accuracy. The proper threshold varies with the signature length change. After all local features in the query image are traversed, we count the accumulated match number for each image in the database. The final retrieval results are based on the verified match numbers.

3.3. Training Patch generation

We carefully design the training patches for the embedding network since the training data is essential to the network performance. Generally all the distorted versions of a patch are regarded as one class. And the two main problems we need to solve are inter-class patches selection and intra-

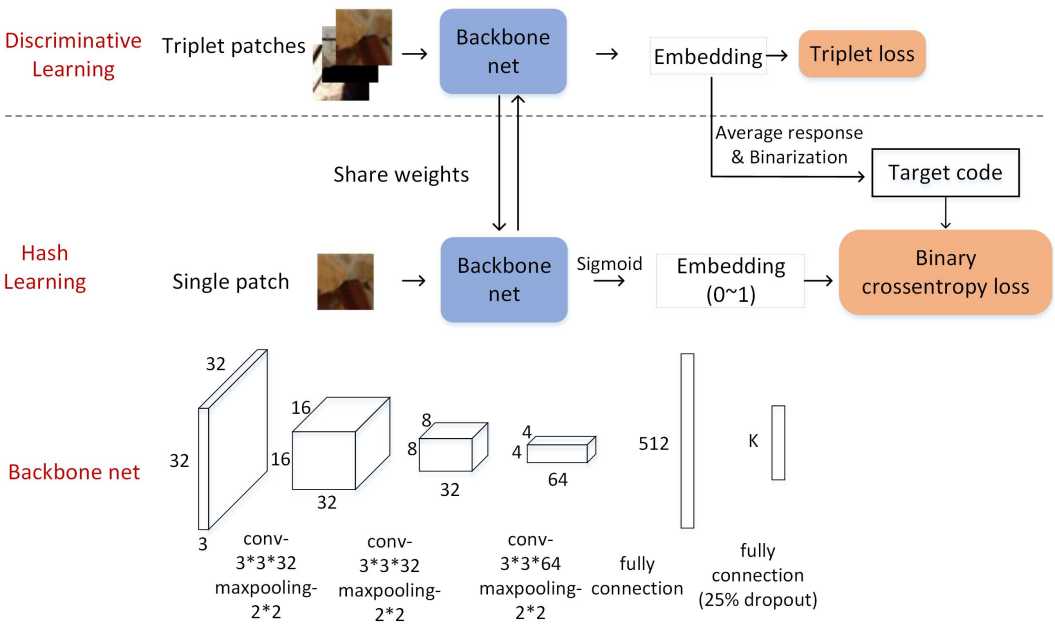

Figure 7: Network structure. The whole training alternates between the Discriminative learning and Hash learning. In discriminative learning stage the network learns how to map image patches into discriminative float features. In the hash learning stage a traversal of training set is first performed and we get the target code for each class. Binary feature can be easily got with a sgn function for testing or retrieving. The convolutional layer in the backbone net refers to the "Conv-BN-RELU" combination.

class samples generation. For the intra-class problem, all the distortion forms which occur in practical use should be considered. We show samples of our processing in Fig. 6. Ordinary transforms such as rotation and adding noise are performed to improve the robustness of the network.

As for the inter-class problem, basically all the interest points detected by the DOG can be chosen, and all we need to do is just sampling. Considering that the learned binary signature should have strong discrimination of patches whose sift features are quantized into the same visual word (Fig. 1), we add these patches from same visual word which have similar structures to our training data. To achieve this, the frequency of a visual word appearing in the whole training images needs to be considered.

The inverse document frequency (IDF) reflects the frequency of a visual word appearing in all the training images. The lower the IDF is, the more frequent a visual word appear in the training images.

$$\text{IDF}(c_i) = log\frac{S}{s_i}, \text{where } s_i = \sum_{j \in \mathcal{G}} \mathbf{1}(\sigma_i^j > 0) \qquad (1)$$

where $\mathbf{G}$ is the training images set, S is the quantity. σ_i^j records the times that visual word c_i appears in image j. Repeatedly appearing visual word has more similar yet non-duplicate structures, and lower IDF. In our approach we select visual words with relative low IDF and select patches from each of these visual words. This strengthens the ability of our embedding network in distinguishing patches from the same visual word. We distort all the training patches as shown in Fig. 6, and training set is then established.

4. Embedding network

In this section we introduce the network which encodes image patches into binary signatures. The whole network includes discriminative learning and hash learning two stages. Fig. 7 shows the framework. We use an alternate training to guarantee the discriminative validity and binary efficiency simultaneously.

4.1. Discriminative learning

The class-based discriminative learning (or metric learning) has been well studied. Recent progress of CNN has proved that a discriminative embedding can be learned with sufficient training samples. Supervised model such as VGG[19] and Resnet[2] trains the classification with images and their labels. The features in middle hidden layers can be viewed as a embedding of the raw image. The unsupervised model such as siamese network[7] and triplet network[18] take image pairs as input and outputs the features such that similar inputs stay close otherwise stay apart. For the discriminative learning in our approach, we adopt the triplet network since it has been proven to be very effec-

tive. The backbone of our network is a traditional CNN model containing three convolution layers and two fully connect layers. We denote the output of the network as $y = \mathcal{F}(x)$. x refers to the image patch in RGB space. Suppose a batch of inputs contain N classes $x_1, x_2, ...x_N$ and M distorted versions for each: $\{x_i^{(1)}, x_i^{(2)}, ...x_i^{(M)}\}_{i=1,2,..N}$. The object of the embedding network is to shorten the distance between near-duplicate patches while enlarge the opposite, i.e. to minimize

$$\sum_{i,j,m_1,m_2} d(y_i^{m_1}, y_j^{m_2})_{i=j} - \sum_{i,j,m_1,m_2} d(y_i^{m_1}, y_j^{m_2})_{i\neq j} \quad (2)$$

The traditional triplet loss function refers to:

$$\mathcal{L} = max(d(a,p) - d(a,n) + margin, 0) \quad (3)$$

in which a is the anchor, p is the positive item and n is the negative item. We adopt the Batch All strategy[3] for the network training. The anchor ranges in all training samples$\{x_i^{(j)}\}i = 1, 2..N, j = 1, 2..M$. And for each anchor we have $M-1$ positive and $(N-1)\cdot M$ negative items. Thus the number of total triplet pairs for a training batch is $MN \cdot (M-1) \cdot (N-1)M$. We only choose the hard and semi-hard pairs for mean value calculation. i.e.

$$loss = \frac{1}{|Q|} \sum_{Q} d(a,p) - d(a,n) + margin \quad (4)$$

where

$$Q = \{(a,p,n)\}_{d(a,p)-d(a,n)+margin>0} \quad (5)$$

After several iterations the network has certain ability to do the classification task. We use this to generate target code and initialize the parameters for the hash learning.

4.2. Hash learning

Hashing method has been proved useful in previous works[10][9]. Generally the hashing methods can be divided into one-step hashing and two step hashing. The two-step hashing methods are mostly used[9][8][26][13]. It first generates the hash code for the giving bits and then learn the hash functions (CNN) using the generated target code. This framework is reasonable since the code to be learned is clear and the only goal of the CNN is just a map function. However there exists a main disadvantage that the binary codes are determined independently of the hash functions. This leads to hard training since we give the target codes in a rigid way and the CNN cannot influence on. For one step approach in some works[10], the binary codes can be produced depending on the CNN outputs with particular regularization, which eases the burden for the CNN.

We combine the advantage of both strategy and make a compromise. Our hash function training is closely connected to the discriminative learning. We use the initial net passed from discriminative learning to get the float features for all the training patches and use (6) to obtain the target binary codes.

$$b_i = 0.5 \cdot (1 + sgn(\bar{y_i})), i = 1, 2, ..N \quad (6)$$

$$\bar{y_i} = \frac{1}{M} \sum_{j=1}^{M} y_i^{(j)} \quad (7)$$

where b_i represents the target code for the ith class which contains K bits. We take the average response of all the distorted versions for the one class. This operation eliminates the case where mutation occurs for some bits. Then we use binary cross entropy loss (8) for the hash function learning.

$$loss = -\frac{1}{K} \sum_{k=1}^{K} (b \cdot log(h(y_k)) + (1-b) \cdot log(1 - h(y_k))) \quad (8)$$

where h is the sigmoid function mapping the embedding into $(0, 1)$

During testing or retrieving the binary signature is obtained by binarizing the output of backbone network using (6).

5. Experiments

5.1. Implementation details

We random select 500 images from VOC2012 as our training source. Then we generate the training patches as described in section 3.3. For the inter-class selection we select the visual words with the 20 lowest IDF and for each visual word we random select 500 patches at most. This generates near 10K patches and we random select 5K for training, 2K for validation. For each patch we generate 200 distorted versions shown in Fig. 6.

During training we set parameters $N = 40, M = 10$ in 4.1, which means the batchsize is 400. $margin$ is 0.5. The bit length is originally set to be $K = 64$ and we adjust it during training. To prevent overfitting we employ a drop out layer with a drop rate of 25% before the final fully connect layer in the backbone net. The whole network is trained on the tensorflow platform with 20 iterations. And for each iteration both discriminative learning and hash learning iterate for 2K steps. We optimize the network with the Adamoptimizer with learning rate 0.001, beta1 0.9, beta2 0.999. Learning rate is reduced every 1K steps with decay rate 0.9.

A view of the 1-norm average value of the embeddings (before sigmoid) changing during network training is shown in Fig. 9. Since for a good embedding learning the variance of the outputs should be large enough and the mean value is supposed to increase by time. During training we find the 1-norm average value increases in the discriminative learning stage while drops in the hash learning stage. And the overall tendency is to increase till it reaches stable.

Figure 8: Near-duplicate samples from the INRIA CopyDays dataset. From left to right the image types: original image, crop, JPEG compression, strongly attacked. The strongly attacked types include print and scan, blur, paint. These distorted versions are regarded as near-duplicate for each other.

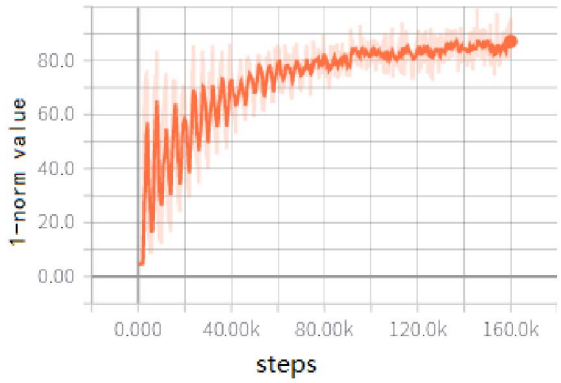

Figure 9: The tendency of embedding 1-norm value during training. The fluctuation shows the different behavior in the two alternate stages. The overall tendency is to grow till it reaches stable

5.2. NDIR performance

We evaluate the effectiveness of our proposed binary feature for the NDIR task on the INRIA CopyDays dataset[5]. Each image in the dataset goes through three different kinds of attacks: crop, JPEG compression, "strong". Fig. 8 gives an impression of these attacks. The "strong" type refers to some irregular attacks such as print and scan, blur, paint. These tasks are more difficult to tackle and are closer to practical use. The whole dataset has 157 original images and about 19 distorted versions for each image. To simulate the large scale application, we follow [22] and random select 100K from Flickr 1M image dataset[4] as distracter images. We use an inverted index file with 100K visual words. For all experiments the local features are first assigned to visual words by sift quantization.

To assess the influence of code length K on retrieval performance, we adjust the code length K during training and produce three different models: '128bit', '64bit' , '32bit'. We test and compare retrieval mAP of the three models.

K	T_s-2	T_s-1	T_s	T_s+1	T_s+2
32	-	0.880	**0.882**	0.865	0.833
64	0.885	0.904	**0.905**	0.897	0.879
128	0.909	**0.916**	0.915	0.908	0.912

Table 1: mAP of our approach using different code length and different threshold. The basic threshold $T_s = K/16$. As code length grows, the retrieval mAP grows correspondingly.

Another parameter related to the performance is the match threshold. During test we observe that the proper threshold appears near $K/16$. We set this value as the basic threshold and denote it as T_s. The best performance may not be achieved exactly at this point. We make slight change during testing and the mAP results are shown in table 1. When code length K is set 128 and threshold is set 7 ($T_s - 1$), our method achieve the best performance.

We compare our proposed method with four other different methods. The *contextual 1*[22] and *contextual 2*[11] are two hand-crafted contextual features. Both of the two contextual features are utilized to reject mismatches immediately. The *rerank*[25] refers to the post-verification method using a spatial coding. The *rerank* is an accelerated method to replace full geometric verification RANSAC[17]. The *baseline* ranks the retrieval results simply by counting matched visual words without subsequent processing. Fig. 10 shows the mAP results.

The query speed and feature storage is essential to practical application. We compare the query time and feature storage of our methods with the others. Fig. 11 shows the results. In Fig. 11 the storage is counted in Byte and we only count a single local feature storage occupation in the inverted index entry. For instance, in baseline method only image ID is stored in the inverted index and thus the storage is 4 bytes (INT). While for our method '32bit' image ID and an 32-bit feature are stored and the storage is 8 bytes.

It can be seen that our approach outperforms the other four methods in mAP. And the query speed is higher due to the fact that we only calculate the Hamming distance. The storage vary with the code length changing and to achieve better results larger storage is needed.

6. Conclusion

In this paper we use a lightweight network to extract local compact binary features for fast reliable near-duplicate patches matching. The networked can be further strengthened by adding distortion forms to the training set when applied to real scenario. We use an alternate training to guarantee that the learned hash code has powerful discrimination ability for the near-duplicate patches. For the NDIR

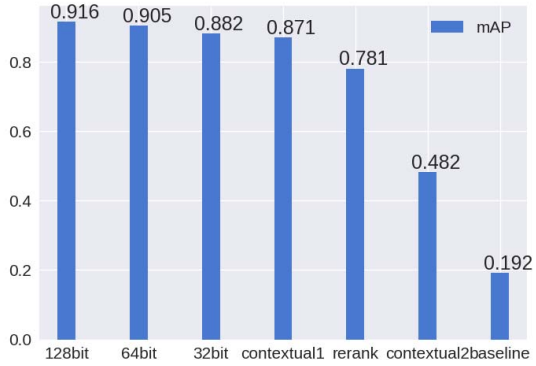

Figure 10: Comparision of mAP of different methods on CopyDays dataset with 100K distracter images. For our methods '128bit', '64bit' and '32bit', we choose the best performance through different thresholds.

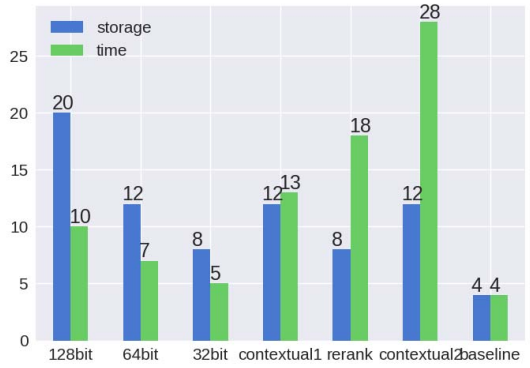

Figure 11: Comparision of query time and storage. Time is counted in seconds for the 100K dataset. Storage is in Byte for a local feature in the inverted list. Our method is fast with an Hamming distance calculation.

task our learned binary feature achieves good performance on both the query results and speed.

References

[1] J. Chen, B. Feng, L. Zhu, P. Ding, and B. Xu. Effective near-duplicate image retrieval with image-specific visual phrase selection. In *2012 19th IEEE International Conference on Image Processing*, pages 1909–1912. IEEE, 2012.

[2] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[3] A. Hermans, L. Beyer, and B. Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017.

[4] M. J. Huiskes, B. Thomee, and M. S. Lew. New trends and ideas in visual concept detection: the mir flickr retrieval evaluation initiative. In *Proceedings of the international conference on Multimedia information retrieval*, pages 527–536. ACM, 2010.

[5] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *European conference on computer vision*, pages 304–317. Springer, 2008.

[6] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *CVPR 2010-23rd IEEE Conference on Computer Vision & Pattern Recognition*, pages 3304–3311. IEEE Computer Society, 2010.

[7] G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2, 2015.

[8] G. Lin, C. Shen, D. Suter, and A. Van Den Hengel. A general two-step approach to learning-based hashing. In *Proceedings of the IEEE international conference on computer vision*, pages 2552–2559, 2013.

[9] K. Lin, J. Lu, C.-S. Chen, and J. Zhou. Learning compact binary descriptors with unsupervised deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1183–1192, 2016.

[10] H. Liu, R. Wang, S. Shan, and X. Chen. Deep supervised hashing for fast image retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2064–2072, 2016.

[11] Z. Liu, H. Li, W. Zhou, and Q. Tian. Embedding spatial context information into inverted filefor large-scale image retrieval. In *Proceedings of the 20th ACM international conference on Multimedia*, pages 199–208. ACM, 2012.

[12] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[13] X. Lu, L. Song, R. Xie, X. Yang, and W. Zhang. Deep hash learning for efficient image retrieval. In *2017 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pages 579–584. IEEE, 2017.

[14] J. Ma, X. Jiang, J. Jiang, J. Zhao, and X. Guo. Lmr: Learning a two-class classifier for mismatch removal. *IEEE Transactions on Image Processing*, 2019.

[15] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[16] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *European conference on computer vision*, pages 143–156. Springer, 2010.

[17] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[18] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[19] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[20] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *null*, page 1470. IEEE, 2003.

[21] Z. Wu, Q. Ke, M. Isard, and J. Sun. Bundling features for large scale partial-duplicate web image search. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 25–32. IEEE, 2009.

[22] J. Yao, B. Yang, and Q. Zhu. Near-duplicate image retrieval based on contextual descriptor. *IEEE signal processing letters*, 22(9):1404–1408, 2014.

[23] L. Zheng and S. Wang. Visual phraselet: Refining spatial constraints for large scale image search. *IEEE Signal Processing Letters*, 20(4):391–394, 2013.

[24] L. Zheng, S. Wang, and Q. Tian. Coupled binary embedding for large-scale image retrieval. *IEEE transactions on image processing*, 23(8):3368–3380, 2014.

[25] W. Zhou, Y. Lu, H. Li, Y. Song, and Q. Tian. Spatial coding for large scale partial-duplicate web image search. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 511–520. ACM, 2010.

[26] B. Zhuang, G. Lin, C. Shen, and I. Reid. Fast training of triplet-based deep binary embedding networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5955–5964, 2016.