

Hash Learning with Convolutional Neural Networks for Semantic Based Image Retrieval

Jinma Guo, Shifeng Zhang, and Jianmin Li^(✉)

State Key Lab of Intelligent Technology and Systems,
Tsinghua National Lab for Information Science and Technology,
Department of Computer Science and Technology, Tsinghua University,
Beijing 100084, China
guojinma@gmail.com, zsffq999@163.com, lijianmin@mail.tsinghua.edu.cn

Abstract. Hashing is an effective method of approximate nearest neighbor search (ANN) for the massive web images. In this paper, we propose a method that combines convolutional neural networks (CNN) with hash learning, where the features learned by the former are beneficial to the latter. By introducing a new loss layer and a new hash layer, the proposed method can learn the hash functions that preserve the semantic information and at the same time satisfy the desirable independent properties of hashing. Experiments show that our method outperforms the state-of-the-art methods by a large margin on image retrieval. And the comparisons with baseline models show the effectiveness of our proposed layers.

Keywords: Hashing · Convolutional Neural Network · Image retrieval

1 Introduction

The amount of web data, images especially, is growing rapidly. How to retrieve images that meet users' requirements from this extremely tremendous data with efficient storage and computation has attracted extensive attentions from academia and industry [3].

Exhaustive nearest neighbor search is intractable. Approximate nearest neighbor search (ANN) can return satisfactory results within logarithmic ($O(\log(n))$) or even constant ($O(1)$) time by organizing data with structures that keep the distance metric. Especially, hashing-based methods [5, 14–16, 19, 24, 26] with lookup tables consume only constant time on a query. The compact codes of hashing can also bring down the demand of storage, and the bitwise operations needed for a query make hashing competent even in the case of exhaustive ranking.

Conventional hashing methods usually take low-level features as input and use shallow models to generate the hash codes. However, the hand-crafted features are fixed and not learnable for further improvements. Recently CNNH [24] gains a great performance boost via deep model to learn hash codes. But this method breaks the learning process into two separate stages. Firstly, pseudo hash codes are learned from images' labels. Then the codes are fixed and used to train a convolutional neural network (CNN) model for later prediction. But some information

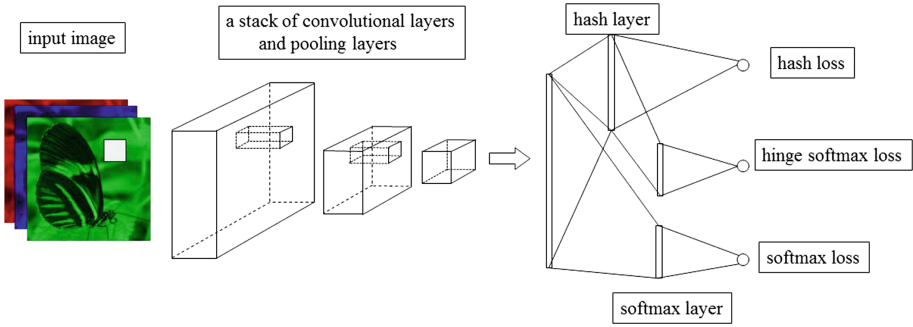


Fig. 1. The architecture of our proposed model in the training stage to learn hash functions. The input size of the model is fixed, and hash layer is fully connected with the prior. When training is done, the two softmax layers can be simply dropped and the outputs of the hash layer are binarized as hash codes.

will be lost in the first stage. Although Lai *et al.* [16] and Zhao *et al.* [26] propose one-stage ranking-based hashing methods respectively, both of which take only ranking as the supervisory information, they do not use the classes information.

In this paper, we propose a novel model that learn deep features and hash functions at the same time. As shown in Fig. 1, the model consists of three parts, which are a stack of convolutional layers, one softmax loss layer for classification, a new proposed hash layer and hinge softmax loss layer for hash code learning. Of the above three parts, the first is used to learn semantic-preserving features, the second is used to encourage the model to learn discriminant features from class labels, while the third part will learn more hashing-like codes. When training is done, the three loss layers will be dropped away and outputs of the hash layer are binarized with 0 to generate the final hash codes. The proposed model is an end-to-end system where feature extraction and hashing are combined.

The specific contributions of our work are as follows:

- (1) we learn hash functions via CNN in the form of multi loss layers
- (2) we introduce the hinge softmax loss layer and a hash layer into hash learning
- (3) as far as we know, our results on the experimental datasets outperform the state-of-the-art.

The remaining is organized as follows: related works are briefly reviewed in Sect. 2. And the methodology of our work is described in Sect. 3. The experiments and discussions are presented in Sects. 4 and 5. Finally, we conclude the whole paper in Sect. 6.

2 Related Work

To generate n -bit code, hashing methods need n hash functions the k_{th} of which generally takes the following form:

$$h_k(x) = \begin{cases} 1 & f_k(x) \geq b_k \\ 0 & f_k(x) < b_k \end{cases} \quad (1)$$

where x is the representation of a data sample, f_k is the hashing function, and b_k is the corresponding threshold. Based on the method to get f_k , hashing can be divided into data-independent methods and data-dependent (or learning-based) methods, of which the latter attempts to capture the inherent distribution of the task domain by learning. And learning-based hashing can be classified into unsupervised and supervised methods according to using annotation information or not.

A typical category of Locality Sensitive Hashing (LSH) [5] uses random projection to construct hash functions. The property of LSH, that samples within small Hamming distance in hash space are more likely to be near in their feature space, makes it very attractive. But the metrics are asymptotically preserved with increasing code length. Thus, to achieve high precision, LSH-related methods require large hash tables.

Unsupervised methods use only unlabeled data as training set, among which are methods such as Kernelized LSH [15], Semantic Hashing [12, 20] and Spectral Hashing [23].

Supervised hashing utilizes human-annotated similarities or labels to get satisfying codes. Supervised Hashing with Kernels (KSH) [19] uses kernel-based model to minimize the Hamming distances of learned hash codes between similar data samples while maximize the distances between dissimilar ones at the same time. Binary Reconstruction Embedding (BRE) [14] learns hash functions by minimizing the differences between original distances of any two samples and the corresponding Hamming distances in hashing space. While initially proposed as unsupervised hashing, BRE can be easily extended to a supervised one by setting similar pairs with distance 0 and dissimilar pairs with distance 1.

These methods are kind of shallow and usually leverage some feature extraction algorithms to get the image representations. But the relationships between samples in semantic space are not maintained in low-level feature space. And even combined with high-level features, the conventional hashing methods are very likely to perform no better than an end-to-end system which learns the feature extractor and hash functions together [26].

On the other hand, explosive interests in computer vision have been attracted to CNN [13] since 2012. Its remarkable successes in kinds of tasks such as object recognition [13, 17, 22], detection [13, 22], image parsing [4] and video classification [10] have narrowed the gap between machine and human vision by a large step.

It has been suggested that the features in deep layers learned from ImageNet possess great capability to represent visual content of images, and can be used for different tasks, such as scene parsing [2], detection [6] and image retrieval [1]. Neural codes [1] uses activations of a fully-connected layer from an ImageNet-pretrained CNN as descriptors of the input image. And then Euclidean distances are computed to measure similarities. When retrained with datasets related to the query field, the retrieval performance can be comparable with the state-of-the-art.

CNNH and CNNH+ [24] take raw image as input, but divide the learning process into two different stages. In the first stage, similarity/dissimilarity matrix is decomposed to get the pseudo binary codes for training images. In the second stage, the raw image pixels and the corresponding binary codes (CNNH+ together with their one-hot binary labels) are fed to a CNN whose objective is to minimize the error between outputs and the target binary codes. But the decomposition stage would bring about extra errors. And the pseudo codes are fixed once the first stage is done, thus are not tunable for further improvement.

Lai *et al.* [16] proposes a deep neural network model to learn the hash functions. And the model’s input is in the form of triplet, i.e., (I, I^+, I^-) , meaning image I is more similar with I^+ than with I^- . The sub-networks for each element image of the triplet share parameters with each other. And the triplet ranking loss function is:

$$\begin{aligned} \text{loss}(x, x^+, x^-) &= \max(0, \|x - x^+\|_2^2 - \|x - x^-\|_2^2 + 1) \\ \text{s.t. } x, x^+, x^- &\in [0, 1]^k \end{aligned} \quad (2)$$

where x , x^+ and x^- are the sub-networks’ outputs of I , I^+ , I^- respectively, and k is the length of x , x^- and x^+ . Zhao *et al.* [26] takes a similar method, but the loss of every triplet is assigned with a weight which is defined by the numbers of shared labels between query image and two result images.

In spite of the above method, [18, 21, 25] assume that good hash codes should also be easily classified by a linear classifier, and take this target into hash learning. In addition to the classification task, [21, 25] penalize the output of embedding functions to make it close to -1 and 1 as much as possible, while [25] also requires the mean of each function to be 0 .

3 Methodology

Given a set of class labels $\mathcal{Y} = \{1, \dots, C\}$ and an image dataset $\mathcal{I} = \{I_1, I_2, \dots, I_N\}$ where each image is associated with one label y_n , our goal is to learn k hash functions which is used to encode images into k -bit hash codes. When using the codes for retrieval, the images sharing same label with the query image will be ranked on top of the result list.

In this paper, we propose a CNN architecture to learn the semantic-preserving hash functions, as shown in Fig. 1. The input image first goes through a stack of convolutional and pooling layers and then arrives at the concatenation layer from where the model branches into two separate paths. Of the two paths, one is the original softmax loss layer and the other is our hash layer and hinge softmax layer.

Normally, suppose that x^l is the output of the l -th layer of a CNN. Then if l -th layer is a softmax layer which is used to predict a vector p of which the c -th element is the probability of class c , the formulation is given by

$$p_c = \frac{\exp(w_c^T x^{l-1})}{\sum_{c'} \exp(w_{c'}^T x^{l-1})} \quad (c = 1, 2, \dots, C) \quad (3)$$

where w_c is the weights related with class c , and x^{l-1} is the output of the prior layer.

3.1 Hash Layer

The hash layer is a fully connected layer which has no nonlinear function inserted. By using fully connection, the hash layer can learn global semantic representations of the input image. And the hash layer's output will be penalized by the following formulation:

$$\text{hash_loss}(x) = \sum_{i=1}^{l_n} ||1 - \text{abs}(g(x_i))||^2 \quad (4)$$

where l_n is the number of neurons in the hash layer. And $g(x_i)$ will take the following form:

$$g(x_i) = \begin{cases} -1 & x_i \leq -1 \\ 1 & x_i \geq 1 \\ x_i & \text{otherwise} \end{cases} \quad (5)$$

This loss can encourage the neurons to generate outputs that distributes less around 0, which will be used as the threshold to binarize the outputs and get the hash codes.

3.2 Hinge Softmax Loss

In hash learning, the target is more a rank problem than classification. It is sufficient to make prediction of ground truth label p_y larger than the rest, while the traditional softmax loss $\text{loss}(y, p) = -\log(p_y)$ can be too harsh. So we define a loss modified from softmax, which takes the following form:

$$\text{hinge_softmax_loss}(y, p) = \begin{cases} 0 & p_y \geq \max(p_{\tilde{y}}) + m \\ -\log(p_y) & p_y < \max(p_{\tilde{y}}) + m \end{cases} \quad (6)$$

where y is the ground truth label, \tilde{y} is the rest of label set, p is the prediction possibilities for every class and m is the slack that controls when the model should be penalized.

By this formulation, for those samples that have been classified correctly by a slack larger than m , the loss will be forced to be 0 and thus back propagate no changes to the learnable parameters. Otherwise, the semantic representations learned by CNN can be not so good, thus the penalization term will be taken. So by this setting, the features of our hash layer can be semantically correct.

When m is set to be 1, because all the prediction possibilities are between 0 and 1, the hinge softmax loss will only execute the lower part and thus becomes conventional softmax loss. And when m is no greater than 0, the loss can be easily stuck in a local minimum, for example when the probabilities of all classes are equal to a certain value.

3.3 The Model

Traditional softmax loss can help in training the networks more discriminant. To combine classification task with hash learning, we branch out from the layer prior to the softmax layer and add a hash layer together with a hinge softmax layer. By this way, the softmax loss in our model performs as an auxiliary classifier like [17].

The overall loss function is given as:

$$\min_W L_1(y, p) + \alpha L_2(x_h) + \beta L_3(y, p) + \lambda \|W\|^2 \quad (7)$$

where W is all the parameters that are to be learned in the network, L_1 is the hinge softmax loss, L_2 is the loss of our proposed hash layer, L_3 the softmax loss for classification and the last term is the weight decay, α and β are hyper-parameters.

By our loss function, the representations learned by the proposed hash layer can preserve the semantic information and be more hashing-like.

Compactness is an important property of hash codes. The binary code generated by every function should be independent with each other and the information carried by the binary bit should be maximized. Compared with [16], we take advantage of dropout’s [8] capability to prevent co-adaptation where a feature detector is only helpful in the presence of several other specific feature detectors. With a dropout layer inserted between the hash layer and hinge softmax loss layer, the neurons of hash layer can be independent of each other.

3.4 Hash Codes

The networks are trained by stochastic gradient descent. When training is done, the two softmax layers can be simply dropped and use the rest architecture to generate the hash codes. When a new image comes, it is first filtered by the model so as to being encoded into a k -dimension vector, and then the vector is binarized into the final hash codes according to Eq. (1) with all b_k set to be 0.

4 Experiments

4.1 Experimental Settings

We compare the proposed model with one data-independent method LSH [5], and four supervised methods BRE [14], KSH [19], CNNH [24] and [16] on two widely used benchmark datasets, i.e., the CIFAR-10 dataset¹ [11, 16] and the Street View House Number (SVHN) dataset². And we will call the model of [16] TRCNNH for short.

For fair comparison, we sample 1000 images from each dataset as query set and another 5000 from the rest for training like [16]. For LSH, all the data except

¹ <http://www.cs.toronto.edu/~kriz/cifar.html>.

² <http://ufldl.stanford.edu/housenumbers/>.

the query set are training set. The 5000 keeping-label samples serve as training set for the other methods. And following [16], images are represented by 512-dim GIST features for non-CNN methods.

As for our proposed method, the 5000 labeled samples in each datasets are divided into training set and validation set by which to find the most suitable architecture and tune the hyper-parameters. Then all the 5000 samples are used to retrain the model from scratch.

The description of an architecture is given in the following way: $3 \times 32 \times 32$ -32C5P2-MP3S2-32C5P0-D0.5-SL10 represents a CNN with inputs of 3 channel of 32×32 pixels, a convolutional layer with 32 filters whose size is 5×5 and 2 paddings around the input maps, a max pooling layer (AP for average pooling) of 3×3 size and stride 2, a 32-filters convolutional layer whose kernel size is 5×5 and have 0 padding around the input maps, a dropout layer whose dropout ratio is 0.5, and finally a softmax loss layer with 10 classes. And our hash layer is always connected with the last hidden layer. Rectifier Linear Unit (ReLU) is used as the nonlinear transformation neurons for all convolutional layers.

Hash lookup and Hamming ranking are two widely used methods to conduct search with hashing [16, 24]. Hash lookup constructs a lookup table with radius r in advance, and all the samples within the radius will be returned as results, thus can decrease the query time to a constant value. However, the number of results returned will dramatically decrease with the code’s length increases. On the other hand, Hamming ranking will traverse the dataset all through at a new coming.

We evaluate the performances on three metrics, i.e. Precision curves within Hamming radius 2, Precision-Recall curves and Precision curves with respect to different returned number with ranking.

Our models are implemented with Caffe [9], an open-source CNN framework. On both datasets, our networks are trained with stochastic gradient descent. The momentum is set to 0.9. Weight decay coefficients for convolutional layers and fully connected layers are separately set to 0.001 and 0.25. For the hyper-parameters in Eq. (7), α and β are decided by validation and fixed at 0.1, while γ decreases from 0.3 to 0 in the whole training stage. The margin in the hinge softmax loss layer is set to 0.1 in our experiments, which won’t hurt too much, and can back propagate at the same time.

Table 1. MAP of Hamming ranking w.r.t different number of bits on two datasets.

Method code length	CIFAR10(MAP)				SVHN(MAP)			
	12 bits	24 bits	32 bits	48 bits	12 bits	24 bits	32 bits	48 bits
Ours	0.611	0.632	0.645	0.641	0.911	0.931	0.934	0.942
TRCNNH [16]	0.552	0.566	0.558	0.581	0.899	0.914	0.925	0.923
CNNH [24]	0.484	0.476	0.472	0.489	0.897	0.903	0.904	0.896
KSH [19]	0.311	0.348	0.353	0.366	0.576	0.631	0.658	0.662
BRE [14]	0.150	0.172	0.174	0.176	0.156	0.168	0.169	0.180
LSH [5]	0.106	0.119	0.121	0.124	0.132	0.143	0.128	0.151

The implementations of BRE [14] and KSH³ [19] are provided by their authors. For LSH, the projections are randomly sampled from a Gaussian distribution with zero-mean and identity covariance to construct the hash tables. And the results of CNNH and TRCNNH are obtained from [16].

4.2 CIFAR-10

The CIFAR-10 dataset is an images collection containing 60,000 color images of 32×32 pixels. All the samples are evenly labeled with 10 mutually exclusive classes, ranging from airplane to bird.

For this dataset, the architecture we choose is a six-conv-layer model. The main branch of the model is $3 \times 32 \times 32$ -32C3P1-32C3P1-MP3S2-D0.5-32C3P1-32C3P1-MP3S2-D0.5-64C3P1-64C3P1-AP3S2-D0.5-SL10. For the convenience of later quotation, we name the third pooling layer “Pool3”.

The MAP of Hamming ranking can be seen on Table 1 and the performance curves are shown in Fig. 2. For all the four experimented code length, i.e. 12, 24, 32 and 48, the MAP of our model surpasses the state-of-the-art by more than 5.9

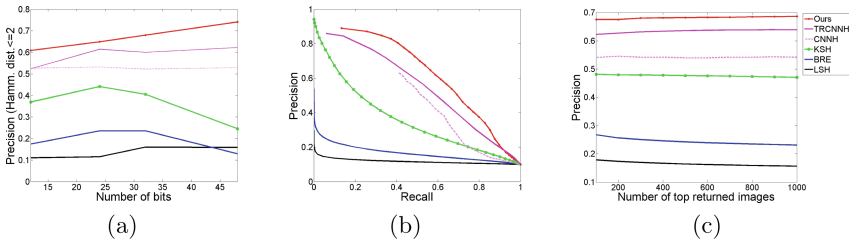


Fig. 2. The results on CIFAR-10. (a) Precision curves of hash lookup within Hamming radius 2. (b) Precision-recall curves of Hamming raking with code’s length of 48 bits. (c) Precision curves with respect to number of returned images of Hamming raking with code’s length of 48 bits.

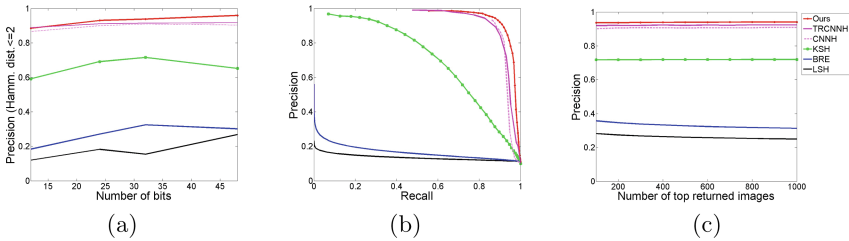


Fig. 3. The results on SVHN. (a) Precision curves of hash lookup within Hamming radius 2. (b) Precision-recall curves of Hamming raking with code’s length of 48 bits. (c) Precision curves with respect to number of returned images of Hamming raking with code’s length of 48 bits.

³ <http://www.ee.columbia.edu/ln/dvmm/downloads/WeiKSHCode/dlform.htm>.

percent. Figure 2(a) witnesses that precision of images within 2 bits Hamming distance returned by our model increases along with the length of hash code. The LSH reported in this paper used random projection which relies on the input feature to maintain the similarity. But because of the semantic gap between low-level feature and content, LSH’s performances are not stable. While BRE and KSH perform better than LSH, they are still worse than CNN based methods. And in Fig. 2(b) the precision-recall curve of our model encloses more spaces than the others, which is consistent with Table 1. Then Fig. 2(c) demonstrates our model’s best performance on Hamming ranking, from 64% to 69%, nearly by 8% beyond TRCNNH.

Comparison Between Different Features. On CIFAR-10, we conduct some other experiments to compare. One is image retrieval using Euclidean distance between corresponding features of query and dataset’s images as similarity, and the features are right the outputs of hash layer (the “L2_fc” method). Another two experiments we undertake are using KSH and LSH with outputs of “Pool3” as descriptors of an input image (the “Pool3_feat”). LSH is also conducted with 512-dimension Gist feature as image descriptions. The network used in this part is the 32-bit network. The results are shown in Table 2.

Surprisingly, we find that L2_fc is not the best, which may be resulted from the denoising property of binarizing. KSH learns hash functions on top of the features extracted from “Pool3”, right the preceding layer of our hash layer. The retrieval precision of “Pool3_feat + KSH” is inferior to our method by a small margin. And the last two LSH-based experiments show that the features learned by CNN are indeed better than Gist.

Comparison with Baseline. We evaluate the performance of our proposed hash layer and hinge softmax loss layer by comparing with three baseline models on CIFAR-10. The first baseline model is just like our model but without the hash loss. The second baseline model is similar with [18], i.e. without the auxiliary path and the hash loss. In addition to the second baseline, the third baseline model replaces the hinge softmax loss with traditional softmax loss. The parameters of the preceding convolutional layers are kept the same. The results are presented in Table 3. Our proposed method performs best on three

Table 2. Ranking precisions of different methods at 500 and 1500 on CIFAR-10

Precision \ Method	Top N		
		500	1500
Ours		68.52%	69.51%
L2_fc		68.30%	66.17%
Pool3_feat + KSH		68.40%	68.88%
Pool3_feat + LSH		58.84%	53.96%
Gist_512 + LSH		16.39%	15.21%

Table 3. Comparison of the proposed model with baseline on CIFAR-10.

Method code length	CIFAR10(MAP)			
	12 bits	24 bits	32 bits	48 bits
Ours proposed model	0.611	0.632	0.645	0.641
Ours without hash loss	0.607	0.621	0.633	0.641
Ours with only hinge_softmax	0.617	0.624	0.633	0.638
CNN with only softmax	0.594	0.629	0.619	0.636

of the four tested code lengths. The third baseline model without any of our modifications performs worst except 24 bits and fluctuate with the code length.

4.3 SVHN

The SVHN dataset consists of 630, 420 color house number images collected from Google Street View images. The data is provided in two formats and the second is used in our experiments. Each data sample is of size 32×32 pixels and annotated with one label from 1 to 10. And the dataset is preprocessed with local contrast normalization, following Goodfellow *et al.* [7].

On this dataset, the main structure is similar to the CIFAR-10 model: $3 \times 32 \times 32$ -128C3P1-128C3P1-MP2S2-D0.5-128C3P1-128C3P1-MP2S2-D0.5-128C3P1-128C3P1-AP2S2-D0.5-SL10.

Similar with the results on CIFAR-10, the MAP values related with the four code lengths of our model on CIFAR10 outperform the state-of-the art. Figure 3(a) shows the precision of returns with hash lookup of radius 2. When using 12-bit codes for retrieval, the precision of our model is just comparable with TRCNNH [16]. But with the increment of code length, the gap between our model and TRCNNH is getting larger. On all the three metrics, our model is the best one.

5 Discussion

Experiments have shown that the precision of our proposed model on both SVHN and CIFAR-10 within Hamming radius of 2 improves with more hash bits, which is consistent with our intuition.

Although KSH uses kernel functions for hashing and our hash layer uses linear functions, the KSH with our CNN-feature performs slightly inferior than our method. The hash layer can be considered as learning-based linear projections, whose inputs are also learnable. During training, the weights and biases in CNN are adjusted to render the filters expressive enough so as to allow the simple classifier on the top perform well. And the raw pixels are at the same time transformed into a description space where similarity relationships are correlated with Euclidean distance.

Unlike CNNH and TRCNNH both of which need the amount of annotations to be square or even cubic of the number of training images, our method trains the CNN with single image whose label indicates the class. So more training data will not burden it too much.

6 Conclusion

As a method of ANN search, interests of many researchers and companies have been attracted by hashing. We propose a new method which can obtain the binary hash code of a given image just by binarizing the outputs of our hash layer, and achieves the best result on both SVHN and CIFAR-10.

In consideration of the fact that all the models use only 5000 samples to train, we can expect an improvement of performance with a larger training set. Besides, a large amount of unlabeled data remains untouched, which can be another key element to enhancement.

Acknowledgments. This work was supported by the National Basic Research Program (973 Program) of China (Nos. 2012CB316301 and 2013CB329403), and the National Natural Science Foundation of China (No. 61332007).

References

1. Babenko, A., Slesarev, A., Chigorin, A., Lempitsky, V.: Neural codes for image retrieval. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014, Part I. LNCS, vol. 8689, pp. 584–599. Springer, Heidelberg (2014)
2. Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Semantic image segmentation with deep convolutional nets, fully connected CRFs (2014). arXiv preprint [arXiv:1412.7062](https://arxiv.org/abs/1412.7062)
3. Datta, R., Joshi, D., Li, J., Wang, J.Z.: Image retrieval: ideas, influences, and trends of the new age. *ACM Comput. Surv. (CSUR)* **40**(2), 5 (2008)
4. Farabet, C., Couprie, C., Najman, L., Lecun, Y.: Learning hierarchical features for scene labeling. *IEEE Trans. Pattern Anal. Mach. Intell.* **35**(8), 1915–1929 (2013)
5. Gionis, A., et al.: Similarity search in high dimensions via hashing. In: Proceedings of 25th International Conference on Very Large Data Bases VLDB 1999, September 7–10, Edinburgh, Scotland, UK, pp. 518–529 (1999)
6. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 580–587. IEEE (2014)
7. Goodfellow, I., Warde-farley, D., Mirza, M., Courville, A., Bengio, Y.: Maxout networks. In: Proceedings of the 30th International Conference on Machine Learning (ICML 2013), pp. 1319–1327 (2013)
8. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors (2012). arXiv preprint [arXiv:1207.0580](https://arxiv.org/abs/1207.0580)
9. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: convolutional architecture for fast feature embedding. In: Proceedings of the ACM International Conference on Multimedia, pp. 675–678. ACM (2014)

10. Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L.: Large-scale video classification with convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1725–1732. IEEE (2014)
11. Krizhevsky, A., Hinton, G.E.: Learning multiple layers of features from tiny images (2009)
12. Krizhevsky, A., Hinton, G.E.: Using very deep autoencoders for content-based image retrieval. In: Proceedings of the European Symposium on Artificial Neural Networks. Citeseer (2011)
13. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems 25, pp. 1097–1105 (2012)
14. Kulis, B., Darrell, T.: Learning to hash with binary reconstructive embeddings. In: Advances in Neural Information Processing Systems 22, pp. 1042–1050 (2009)
15. Kulis, B., Grauman, K.: Kernelized locality-sensitive hashing for scalable image search. In: IEEE 12th International Conference on Computer Vision, pp. 2130–2137. IEEE (2009)
16. Lai, H., Pan, Y., Liu, Y., Yan, S.: Simultaneous feature learning and hash coding with deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2015)
17. Lee, C.-Y., Xie, S., Gallagher, P., Zhang, Z., Tu, Z.: Deeply-supervised nets. In: Proceedings of the International Conference on Artificial Intelligence and Statistics, pp. 562–570 (2015)
18. Lin, K., Yang, H.-F., Hsiao, J.-H., Chen, C.-S.: Deep learning of binary hash codes for fast image retrieval. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 27–35 (2015)
19. Liu, W., Wang, J., Ji, R., Jiang, Y.-G., Chang, S.-F.: Supervised hashing with kernels. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2074–2081. IEEE (2012)
20. Salakhutdinov, R., Hinton, G.: Semantic hashing. *Int. J. Approximate Reasoning* **50**, 969–978 (2009)
21. Shen, F., Shen, C., Liu, W., Shen, H.T.: Supervised discrete hashing. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. IEEE (2015)
22. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition (2014). arXiv preprint [arXiv:1409.1556](https://arxiv.org/abs/1409.1556)
23. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing. In: Advances in Neural Information Processing Systems 21, pp. 1753–1760 (2008)
24. Xia, R., Pan, Y., Lai, H., Liu, C., Yan, S.: Supervised hashing for image retrieval via image representation learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 2156–2162 (2014)
25. Yang, H.-F., Lin, K., Chen, C.-S.: Supervised learning of semantics-preserving hashing via deep neural networks for large-scale image search (2015). arXiv preprint [arXiv:1507.00101](https://arxiv.org/abs/1507.00101)
26. Zhao, F., Huang, Y., Wang, L., Tan, T.: Deep semantic ranking based hashing for multi-label image retrieval. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1556–1564 (2015)