BINARY HASHING USING SIAMESE NEURAL NETWORKS

Abin Jose, Shen Yan and Iris Heisterklaus

Institut für Nachrichtentechnik, RWTH Aachen University, Aachen, Germany
jose@ient.rwth-aachen.de, shen.yan@rwth-aachen.de, heisterklaus@ient.rwth-aachen.de

ABSTRACT

With the growth in multimedia data, it is the need of the hour to have methods for efficient storage and quick retrieval. In this work, we propose an approach for learning binary codes for fast image retrieval. We use a siamese architecture with two parallel feed forward branches but with a shared weight for the generation of binary codes. The training data is divided into similar and dissimilar pairs. The network tries to learn the weights such that it minimizes the distance between similar image pairs and maximizes the distance between dissimilar image pairs. The binary codes are formed by squashing the neural network output through a sigmoid activation function. The training with sigmoid hashing constrains the output of each node in the final fully connected layer to either 0 or 1. We have compared the retrieval performance of our approach with other state-of-the-art hashing methods and our method shows significant improvement.

Index Terms— Binary hashing, Siamese neural networks, Image retrieval, Similar image pairs, Dissimilar image pairs.

1. INTRODUCTION

In content-based image retrieval systems (CBIR) [1], the visual content of the images is the main retrieval criterion. This visual content is represented by local feature descriptors like SIFT [2] descriptors which describe each image uniquely. Global descriptors such as Bag of Visual Words (BoVW) [3], Vector of Locally Aggregated Descriptors (VLAD) [4], and Fisher Vectors (FV) [5] are formed from these local descriptors and using a similarity measure, images from the database can be ranked and retrieved.

However, with the advances in the area of artificial intelligence, we are now capable of training neural networks which acts as feature extractors. Another reason for the recent advances in this area is the availability of powerful Graphical Processing Units (GPUs) capable of training neural networks quickly. The vast amount of training data needs to be efficiently stored and retrieved and this led to active research for generating binary codes which uniquely store and represent the images.

Several hashing approaches have been proposed recently to efficiently generate binary codes. They are broadly classified into supervised and unsupervised approaches. In unsupervised approaches, the training data is not labeled. One of the earlier ideas in this category is Locality Sensitive Hashing [6] which tries to maximize the probability such that similar data have similar hashes. However, it generally requires longer codes for good performance. Spectral hashing [7] is another popular technique which tries to minimize the Hamming distance between image pairs. Another method, called Iterative Quantization [8] minimizes the quantization error of mapping a zero-centered data to the vertices of a zero-centered binary hypercube. In contrast to unsupervised approaches, supervised methods use the labeled dataset. Methods such as binary reconstructive embedding [9] generate similarity preserving hashing in kernel space. The aforementioned hashing approaches use hand-crafted features.

Recently Convolutional Neural Network (CNN) based hashing approaches have gained popularity. Alexnet [10] uses the features from the seventh layer (fully connected layer) of CNN and achieves good retrieval accuracy. Babenko et al. [11] compress the CNN features and report good retrieval performance. It is noted that, in these approaches, similarity comparison is performed in Euclidean space which is inefficient since it is more time consuming.

Currently, techniques which generate binary codes for fast image retrieval with Hamming distance as the similarity criterion is quite an active research area. The main reason is that it is much faster to measure image similarity in Hamming space than the Euclidean space. Lai et al. [12] uses a triplet architecture and uses a divide-and-encode module for generating binary codes. In the model proposed by Lin et al. [13], a latent layer is introduced for generating binary codes in the conventional feed forward architecture. However, they use transfer learning to effectively extract the features. [14] as well uses triplet model and enforces additional constraints to the loss function to generate efficient binary codes. Liu et al. [15] uses the siamese model for generating binary codes. They introduce an additional regularizer to the loss function whose value heavily influences the binarization step. [16] proposes a method in which the output of the penultimate layer of a feed forward neural network is constrained to $\{1, -1\}$ and are used as the binary codes. However, here the hash function is defined by multiple hierarchichel layers of non-linear and linear transformations. The pre-trained parameters of convolutional layers and fully connected layers are used to intialize the CNN in [17] followed by the hashing layer. The use of pre-trained model makes this model dataset specific.

Our contribution in this paper is two-fold. Firstly, we show that we can generate efficient binary codes by using a siamese model. This model aggregates the feature vectors for similar images together and pushes apart the feature vectors of dissimilar images. After training, the neural network is capable of capturing the image semantics. Here, the fully connected layers form the corresponding feature vectors. We employ a non-linear activation function such as the sigmoid function after the final fully connected layer which acts as a hashing function. The length of the binary codes can be varied by changing the number of nodes in the final fully connected layer. Conventional hashing approaches use pretrained models and generate the image pairs offline. We further propose a novel training approach on how to effectively train the model by generating similar and dissimilar pairs of images during the training phase.

Organization of the paper: In Section 2, we explain the architecture of siamese neutral networks. In Section 3, the new training strategy we have used is elaborated. Experimental results are discussed in Section 4 and concluding remarks are drawn in Section 5.

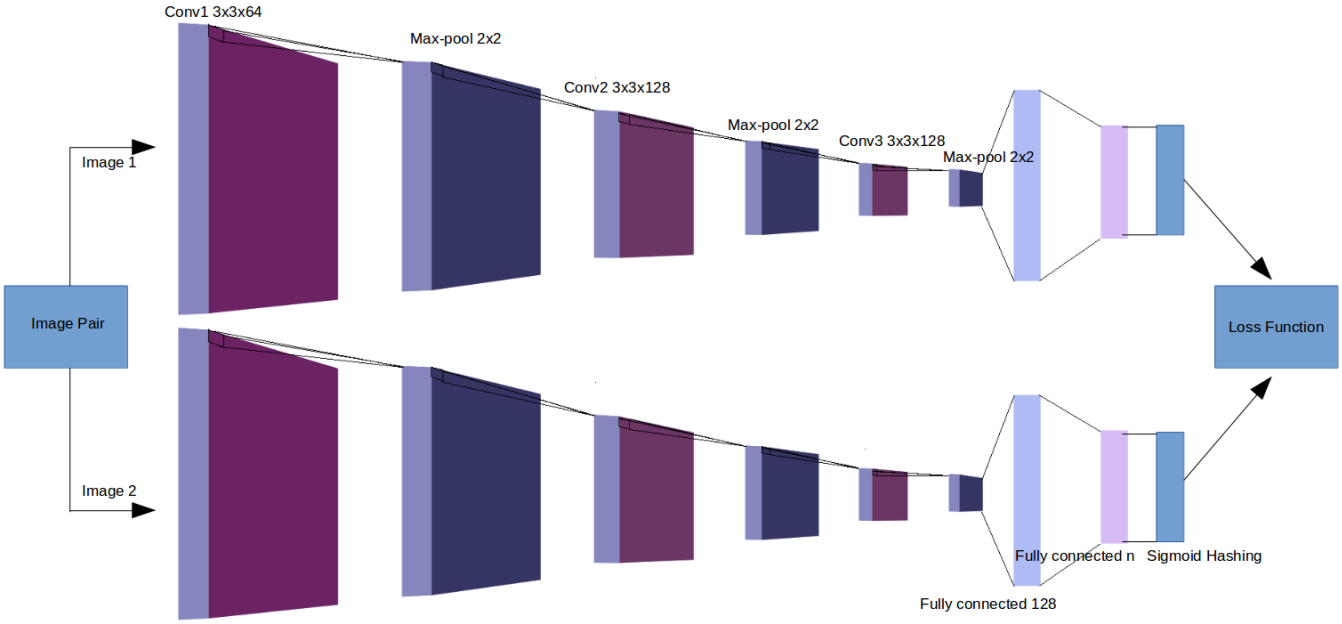


Fig. 1: Architecture of the siamese neural network. The input is fed to the neural network as image pairs. The output of the fully connected layer is passed through the sigmoid function. The distance between the output feature vectors is calculated inside the loss function.

2. SIAMESE NEURAL NETWORKS

The concept of siamese neural networks in the area of artificial intelligence was first coined by LeCun et al. [18] with application towards face recognition. The main motivation of using a siamese model for binary hashing is that, they are capable of learning image similarity or dissimilarity which is basically the desired feature of a retrieval system. Similar images must have similar feature embedding and dissimilar images must be further away in the feature space. A siamese model tries to aggregate similar feature vectors together and dissimilar feature vectors are pushed beyond a margin.

A siamese neural network consists of two feed forward branches but the weights are shared between the two branches. The architecture of the siamese neural network used in our experiment is shown in Figure 1. It consists of three convolutional layers. Each convolutional layer is followed by Rectified Linear Unit (ReLU) nonlinearity and max-pooling operation. Each branch of the siamese model acts as a feature extractor. There are two fully connected layers. The final fully connected layer contains a variable number of nodes n which is the number of bits in the binary code generated. The output of this fully connected layer is passed through the sigmoid layer which squashes the output of each node to either 0 or 1.

The loss function module tries to compute the distance D, between the feature vectors extracted from the 2 branches of the network. The loss function which the network tries to minimize is defined as:

$$\text{Loss}\ (D, Y) = Y \quad \max\ (\text{margin} - D, 0) + (1 - Y)\ D, \quad (1)$$

where Y is the label indicating similar image pair or dissimilar image pair. Y equal 1 means the image pair is dissimilar and Y equal 0 means the image pair is similar.

The first term in Equation 1 penalizes dissimilar image pairs whose distance is less than the margin. Once the distance between dissimilar pairs is learned beyond a value greater than the margin, its contribution to loss function is always zero. The second term tries to minimize the distance between similar image pairs. After the network is trained with sufficient amount of training data, the loss function should theoretically approach 0. It indicates that all the similar image pairs have a distance of 0 between the codes. Furthermore, the dissimilar image pairs are pushed beyond the margin thus contributing $\max\ (\text{margin} - D, 0) = 0$ to the loss function.

The total loss function for N training pairs is defined as: $\mathcal{L} = \sum_{i=1}^{N}(\text{Loss}(D_i, Y_i))$. The goal is to minimize $\mathcal{L}$ using the stochastic gradient descent approach. Once the training is completed or the loss function reaches the minimum, the feature space will be such that the feature vectors of similar images aggregate together in the n-dimensional feature space. The network weights are stored after the training is completed. For generating the binary code for an image, the image is propagated through one branch of the siamese network and the activation of the final layer corresponds to the binary code for the image under consideration.

3. GENERATION OF IMAGE PAIRS FOR TRAINING

The main requirement with the siamese architecture is the need for paired training data classified into two classes - similar pairs and dissimilar pairs. In general, the training data is prepossessed as similar and dissimilar pairs beforehand. This has mainly two drawbacks. The first limitation is that the number of dissimilar pairs is always much greater than the number of similar pairs. This means the training data is unbalanced. For instance, training data with C object classes with N images in each class can generate $C \times \binom{N}{2}$ similar pairs and $N^N \times \binom{C}{2}$ dissimilar pairs. The second limitation is that, since the training pairs are generated offline and stored before hand, there is less randomness in the data. Neural nets tend to be able to learn more effectively when more randomness is introduced in the data. This means that in each iteration, the network should be ca-

Algorithm 1 : Generation of similar and dissimilar pairs

Input: Training data X $\{x_i \mid$ i = 1,2,..,N$\}$. Labels L $\{l_i \mid$ i= 1,2,..,C$\}$.
Indexing Matrix I $\{i_c \mid$ c = 1,2,..,C$\}$, each row of I represents the indices of the corresponding class.
Output: Random similar pairs $\{x_i^a, x_i^s\}$ and hard dissimilar pairs $\{x_i^a, x_i^d\}$.

```
for i = 1 to batchsize -1 do
    Select anchor images x_i^a = x_i.
    Randomly sample a similar image x_i^s from X{I{l_i}}.
    for j = i + 1 to batchsize do
    Select all dissimilar x_j^d within the mini-batch.
    Generate the hardest dissimilar x_j^d = arg min_{x_j^d} ||f(x_i^a) − f(x_j^d)||^2.
    end for
end for
```

pable of seeing new random image pairs. We took care of both this aspects in training and tried to generate image pairs on the fly during the training phase.

The training for neural networks is done in mini-batches. In each mini-batch, similar and dissimilar pairs are generated. At first a random image (anchor image) is selected from the mini-batch. Now, the simplest idea for creating dissimilar pair might be to randomly choose an image with different label than the anchor image. However, after few training iterations, the distance between dissimilar pairs selected this way might be already beyond the margin. Thus its contribution to loss function will be 0. Therefore, for better learning of discriminative feature vectors, choosing informative pairs is crucial.

In our training algorithm, for each anchor within the mini-batch, we select a random image, x_i^s which has similar label as the anchor image. Let $f(x_i)$ corresponds to feature vector for image x_i. Then we generate a hard dissimilar image, x_i^d which is having a distance $\|f(x_i^a) - f(x_j^d)\|^2$ less than the margin. This gives the network an opportunity to learn and push this dissimilar pair beyond the margin. The generation of similar and dissimilar pairs is explained in Algorithm 1.

In practice, it is highly computationally complex to compute the dissimilar pairs across the whole training set during the training phase. Instead, we sample the dissimilar pairs based on each mini-batch and compute the *argmin* on a subset of the data. We observe that in this manner, the training error decreases faster.

4. EXPERIMENTAL RESULTS

We have used CIFAR-10 [19] dataset for the evaluation purpose. The dataset consists of 60,000 images of size 32×32 in 10 different categories. The dataset was split into training data and test data with 50,000 images in the training data and 10,000 images in the test data. The main evaluation metric used is the Mean Average Precision (MAP). The retrieval performance was measured for different number of bits of the binary code word by changing the number of nodes in the final fully connected layer of the neural network.

4.1. Siamese model and training details

The siamese neural network used for training is shown in Figure 1. The network consists of 3 convolutional layers with 3×3 convolutional filters with a stride of 1. The number of filters in each layer are

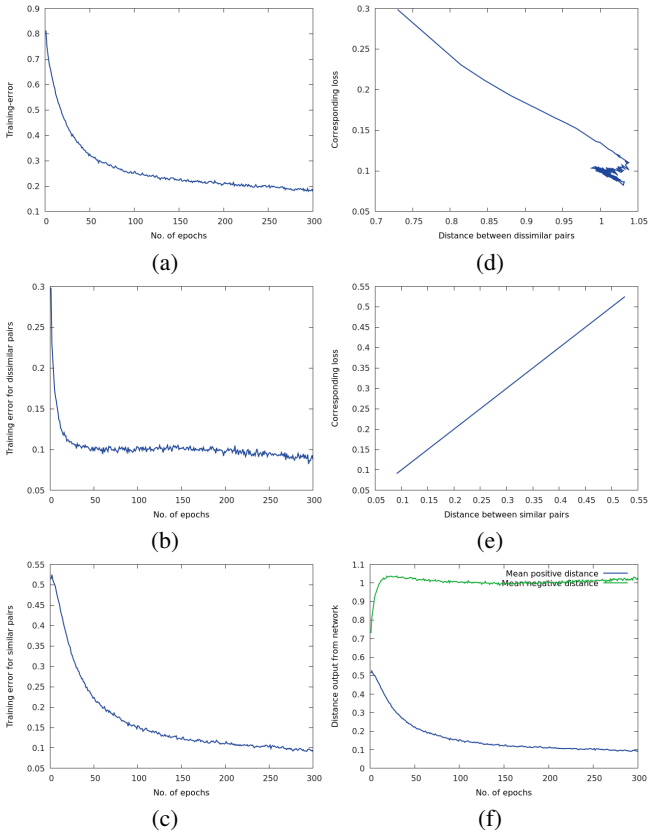


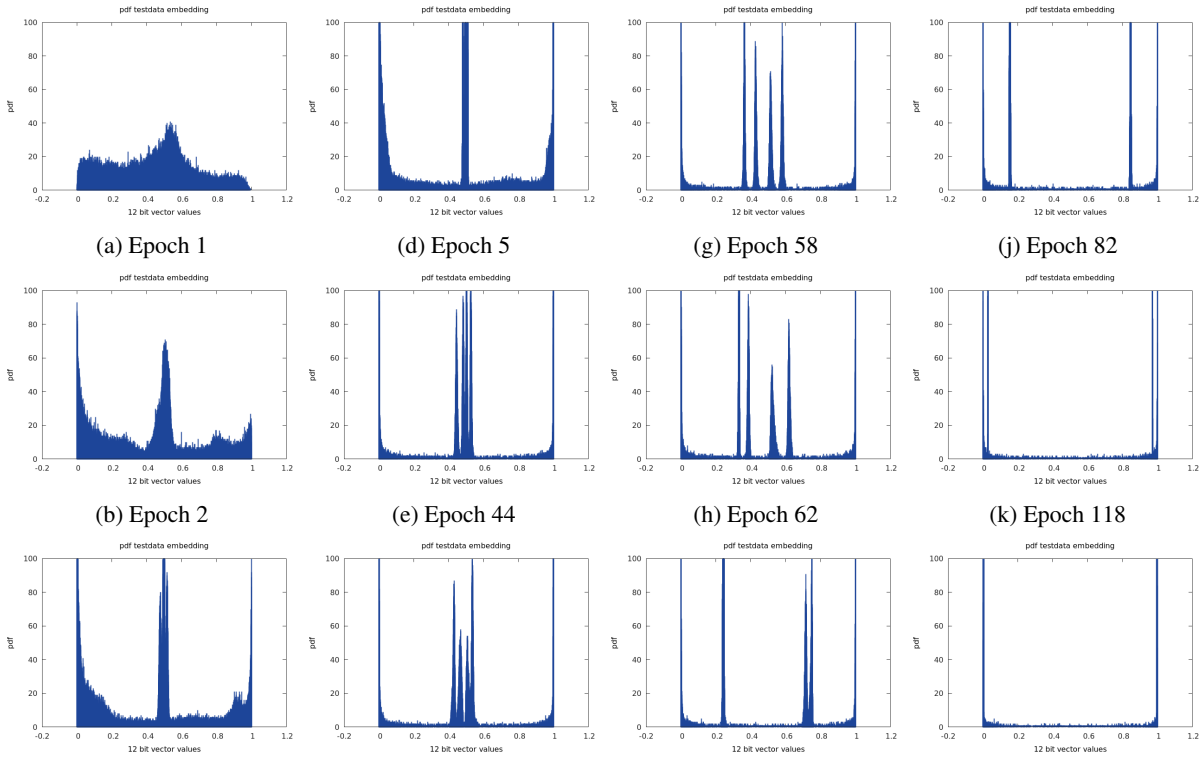
Fig. 2: (a) Total training error, (b) Training error for dissimilar pairs, (c) Training error for similar pairs, (d) Distance between dissimilar pairs vs the corresponding loss, (e) Distance between similar pairs vs the corresponding loss, (f) Change in mean distance of similar pairs and dissimilar pairs as the training proceeds.

64, 128 and 128 respectively. The convolutional layers are followed by ReLU non-linearity and max-pooling operation. The pooling is performed over 2×2 windows with a stride of 2. There are 2 fully connected layers following the convolutional layers. The first fully connected layer contains 128 nodes and the second fully connected layer contains n nodes where n is the number of bits in the binary code generated.

The training data is generated as explained in Algorithm 1. The weight layers are initialized with random initialization. During training, the batch size has been set to 50, learning rate 0.01, margin 1 and momentum 0.9. The network has been trained for 300 epochs. The model has been implemented using the Torch [20] deep learning library. The GPU used is GeForce GTX Titan X.

4.2. Comparision with state-of-the-art hashing methods

We have compared our results with state-of-the-art hashing methods. The test data is considered as query images, making 1,000 query images per class. The database on which the query is made is the training dataset with 50,000 images. The results for varying bit size n is summarized in Table 1. We compare our results with Supervised Hashing with Binary Deep Neural Networks (SH-BDNN) [16], Deep Regularized Similarity Comparision Hashing (DRSCH)



(a) Epoch 1 (d) Epoch 5 (g) Epoch 58 (j) Epoch 82

(b) Epoch 2 (e) Epoch 44 (h) Epoch 62 (k) Epoch 118

(c) Epoch 4 (f) Epoch 48 (i) Epoch 74 (l) Epoch 219

Fig. 3: The probability distribution of the 12 output nodes of the neural network plotted at random training epochs. The learning is quite fast in first few epochs as the training error decreases fast in the initial epochs. It basically depands on the learning rate chosen for the network.

[14] and Deep Semantic Ranking Hashing (DSRH) [17]. For measuring the retrieval performance, we have used the standard experimental settings and MAP is measured at $1{,}000^{\text{th}}$ retrieved image.

Table 1: MAP@1000 in % for CIFAR-10 dataset. The results of [16], [14], [17] are cited from [16].

No. of bits	16	24	32	48
Our method	62.61	64.67	**66.35**	**67.19**
SH-BDNN [16]	**64.3**	**65.21**	66.22	66.53
DRSCH [14]	61.46	62.19	62.87	63.05
DSRH [17]	60.87	61.33	61.74	61.98

We can observe that as the number of bits increase, we have better retrieval accuracy. The improvement in retrieval results can be mainly attributed to the training approach proposed in this paper. This proves that the learning of neural networks, heavily depends on the randomness introduced in the training data. Our results prove that without data augumentation or without additional regularizers in the loss function, better codes can be generated provided we generate training pairs which the network has not seen before.

Figure 2 (a) shows how the training loss decreases as the training proceeds. The training loss can be split into 2 parts. Figure 2 (b) shows the training loss for dissimilar pairs and Figure 2 (c) shows the corresponding loss for similar pairs. We can observe that both these losses decrease as the training proceeds which is the expected behaviour. Figure 2 (d) shows how the mean distance between dissimilar pairs change as the network trains. As the training loss decreases, the average distance of dissimilar pairs settle around the margin. Also as the training loss decreases, mean distance of similar pairs approach to 0 as shown in Figure 2 (e). Figure 2 (f) again shows that, as the training proceeds, the mean distance of dissimilar pairs goes beyond the margin 1 and the mean distance of similar pairs approach 0 thus pushing the total loss function to a minimum.

Figure 3 shows another interesting aspect of our experiments where we plot the total distribution of the n output nodes for all the training images. (The behaviour is similar for each epoch on the train data as well which proves that there is no overfitting). We have a Laplacian distribution at the first epoch (Figure 3(a)). As the training proceeds, we can observe that the distribution start to peak around 0 and 1. Once the training is completed, the distribution is completely binarized as shown in Figure 3(l).

5. CONCLUSIONS

We propose a method for generation of binary codes for images using siamese architecture. The main contributions of this work is two-fold. We use the sigmoid layer which helps the network to learn binary codes by itself. We also propose a method for the online generation of training pairs by which the network get acquainted with more random pairs of similar and dissimilar images. We have shown that the bit size, n for the binary codes generated can be varied relatively easily by changing the number of nodes in the final fully connected layer. The experimental results show that with higher number of bits in the binary codes, we have better representation of the images which in turn increases the retrieval performance.

6. REFERENCES

[1] Arnold W M Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh Jain, "Content-based image retrieval at the end of the early years," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 12, pp. 1349–1380, 2000.

[2] David G Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, 1999, vol. 2, pp. 1150–1157.

[3] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray, "Visual categorization with bags of keypoints," *ECCV Workshop on Statistical Learning in Computer Vision*, vol. 1, no. 1-22, pp. 1–2, 2004.

[4] Hervé Jégou, Florent Perronnin, Matthijs Douze, Javier Sanchez, Pablo Perez, and Cordelia Schmid, "Aggregating local image descriptors into compact codes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 9, pp. 1704–1716, 2012.

[5] Jorge Sánchez, Florent Perronnin, Thomas Mensink, and Jakob Verbeek, "Image classification with the fisher vector: Theory and practice," *International Journal of Computer Vision*, vol. 105, no. 3, pp. 222–245, 2013.

[6] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al., "Similarity search in high dimensions via hashing," in *Proceedings of the international conference on Very Large Data Bases*, 1999, pp. 518–529.

[7] Yair Weiss, Antonio Torralba, and Rob Fergus, "Spectral hashing," in *Advances in neural information processing systems*, 2009, pp. 1753–1760.

[8] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 12, pp. 2916–2929, 2013.

[9] Jonathan Long, Evan Shelhamer, and Trevor Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[11] Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky, "Neural codes for image retrieval," in *European conference on computer vision*. Springer, 2014, pp. 584–599.

[12] Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan, "Simultaneous feature learning and hash coding with deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3270–3278.

[13] Kevin Lin, Huei-Fang Yang, Jen-Hao Hsiao, and Chu-Song Chen, "Deep learning of binary hash codes for fast image retrieval," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2015, pp. 27–35.

[14] Ruimao Zhang, Liang Lin, Rui Zhang, Wangmeng Zuo, and Lei Zhang, "Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification," *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 4766–4779, 2015.

[15] Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen, "Deep supervised hashing for fast image retrieval," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2064–2072.

[16] Thanh-Toan Do, Anh-Dzung Doan, and Ngai-Man Cheung, "Learning to hash with binary deep neural network," in *European Conference on Computer Vision*. Springer, 2016, pp. 219–234.

[17] Fang Zhao, Yongzhen Huang, Liang Wang, and Tieniu Tan, "Deep semantic ranking based hashing for multi-label image retrieval," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1556–1564.

[18] Sumit Chopra, Raia Hadsell, and Yann LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005, vol. 1, pp. 539–546.

[19] Alex Krizhevsky and Geoffrey Hinton, "Learning multiple layers of features from tiny images," 2009, Citeseer.

[20] S Sonnenburg, "Nips workshop on machine learning open source software," 2006.