Deep Feature Based on Convolutional Auto-Encoder for Compact Semantic Hashing

Jun Wang\textsuperscript{a}, Jian Zhou\textsuperscript{b}, Liangding Li, Jiapeng Chi, Feiling Yang and Dezhi Han
University of Central Florida, USA
\textsuperscript{a}jun.wang@ucf.edu; \textsuperscript{b}jian.zhou@ucf.edu

Abstract. For content-based image retrieval, a good presentation is crucial. Nowadays, as deep learning models can be used to generate an excellent presentation, it has been extensively investigated and widely used in research systems and commercial production systems. However, the deep representation (deep feature) is still too large. Compared with directly using deep representation, binary code can reduce significant storage overhead. Meanwhile, the bit-wise operations for binary code can dramatically fasten the computation. There exist some schemes used to convert the deep feature to binary code, but all of them directly applied the last layer of the connection layers, which exhibit global feature and discriminating features. To achieve deep generative feature and avoid destroying the image locality, we aim to construct the binary hash code based on convolutional auto-encoders. Namely, we use the generative model to transform the local feature to binary code. The training process of our proposed model is decomposed into three stages. Firstly, the convolutional layers are trained using convolutional autoencoders, followed by the fully-connected layers training using Restricted Boltzmann Machine. Thirdly, we deploy a supervised similarity learning algorithm to learn close code for similar images.

1. Introduction
An efficient and high-efficient representation is critical for CBIR. One kind of scheme exploits the low-level features for image retrieval. In intuition, these plans are derived from the document retrieval using bag-of-word and inverse index. Namely, the image is converted into a bag of visual words, then searched by reverse index. However, these kinds of schemes are prone to miss the semantic features. If we apply the document retrieval, the retrieval time is about $O(N \ast v)$ using naive manner, If we use the TF-IDF for the image retrieval using visual terms or bag-of-words, the retrieval time will reduce into $O(N \ast v)$. With the rapid development of deep learning, deep features extracted by deep learning models, such as CNN and DAE, have been applied for content-based image retrieval.

However, the high-dimension deep feature introduces a significant amount of storage overhead and network overhead. For content-based image retrieval, short-hash codes are crucial. Compared to matching pixel intensities directly or real-valued high dimension feature matching, short binary codes can conserve more storage space and communication overhead. Auto-encoder have become a popular model because of its ability to produce a better representation as input to a supervised model than the raw input. It has successfully been applied to many domains, such as classification, clustering, object recognition. There emerge some researches [1] [2] [3] [4] [5] [6] [7] [8] which focus on exploiting deep learning to construct compact binary codes. These schemes can be classified into two kinds. One kind is based on convolutional neural network, such as [1] [2] [3] [4] [5] [6], which utilize high-level features, i.e. deep feature, to learn binary-like hash code. In order to hash the high-dimension deep feature into
binary hash code, these schemes directly exploit the last layer in the fully connected layers. However, the deep feature in the fully-connected layer exhibits discriminant characteristics, which perform worse than generative feature for content-based image retrieval. Another kind is based on deep generative models [7] [8]. However, these models cannot capture the local image features. To achieve deep generative feature and avoid destroying the image locality, we aim to construct the binary hash code based on convolutional auto-encoders. Because convolutional auto-encoders consists of too many layers, we decomposed the training process into convolutional layers and fully-connected layers. In overall, the training process is accomplished by three stages. Firstly, the convolutional layers are trained using convolutional auto-encoders, followed by the fully-connected layers training using Restricted Boltzmann Machine. Thirdly, we deploy a supervised similarity learning algorithm to learn close code for similar images.

The main contributions are shown as follows. (1) To our knowledge, we are the first to apply the convolutional auto-encoders to construct binary code. (2) We design a training algorithm for our proposed model. (3) We evaluate and compare our mean average precision (mAP) of Hamming ranking i.e. the quality of retrieval for different number of bits on MNIST, CIFAR10 and NUS-WIDE datasets with state of art hashing algorithms.

2. Design

The text of your paper should be formatted as follows: Compared with directly using deep representation, binary code can reduce significant storage overhead. Meanwhile, the bit-wise operations for binary code can dramatically fasten the computation. There exist some schemes used to convert the deep feature to binary code, but all of them directly applied the last layer of connection layers, which represents exhibits global feature and discriminating features. In this paper, we use the generative model to transform the local feature to binary code.

2.1. CAE for Content-based image retrieval

2.1.1 Motivation. In CNN based image classification, the low-level layer exhibits the local features, and high-level layers show the global features. There are two kinds of index schemes for content-based image retrieval. The first kind is a scheme for CBIR based on low-level features. It can be implemented using bag-of-word and inverse index. The second kind is a scheme for CBIR based on high-level features; it can be implemented by hash code.

We aim to analyse the impact of low-level and high-level features on the effectiveness on CBIR. First, the convolutional autoencoder can reduce the dimension while preserving the local image structure. Second, the auto-encoder can reduce the dimension with preserving the semantic similarity. Auto-encoders allow to represent data in low-dimensional spaces. If we can reconstruct the data from such low dimensional spaces, it means that these latent spaces contain enough high-level semantic information.

2.1.2 Main idea. The convolutional neural network consists of convolutional layers, pooling layers, and fully-connected layers. It can generate good representation using local connection and shared weight. However, the binary code generated by CNN only consider the global features. There exist some papers [7] [9], which utilize the stacked auto-encoders to generate binary hash code. However, the stacked auto-encoders model does not consider the local field. As shown in Figure 1(c), this is the model we aimed to build. We use convolutional encoders. Unlike the CNN, it is a reconstruction model. Unlike stacked auto-encoders, it employs the convolutional operations and pooling operations for the deep modes. As shown in Figure 1(c), we aimed to analyze the impact of low-level and high-level features on the effectiveness on CBIR. First, the convolutional auto-encoder can reduce the dimension while preserving the local image structure. Second, the auto-encoder can reduce the dimension with preserving the semantic similarity.
2.2. Auto-encoders and convolutional auto-encoders

Auto-encoder have been widely applied in dimension reduction, feature extraction and so on. Auto-encoder usually consists of three layers: an input layer, hidden layer, output layer. As shown in Figure 1(a), the training or inference of Auto-encoder includes two parts: encoder and decoder. The encoder means to map the input vector \( x \) into the hidden vector \( h \) using a deterministic function. Namely, the encoder process can be formulated as follows.

\[
h = \sigma(Wx + b)
\]  

(1)

where the \( \sigma \) represents the sigmoid function (e.g. tanh or logistic function). The parameters \( W \) and \( b \) represent the weight and bias factor between two layers. In contrast, the decoder means to map the hidden vector \( h \) into the output vector \( y \) using another deterministic function. The parameters for the encoder and decoder are usually constrained into the same. Namely, the weight for the decoder is the inverse matrix of \( W \).

Because the auto-encoder always learns identity mapping, this problem can be handled by some models, such as Restricted Boltzmann Machine, Sparse code, Denoising Auto-Encoder and so on. Denoising Auto-Encoder add some noise to the input vector. For example, some units in the input vector are corrupted as shown in the Figure 1(b).

![Auto-encoder and Denoising Auto-encoder](image)

(a) Auto-encoder  
(b) Denoising Auto-encoder  
(c) Convolutional Auto-Encoder

Figure 1: Auto-encoder and Denoising Auto-encoder

However, the auto-encoder does not consider the image structure. The local structure can be captured by convolutional operations and pooling operations. As shown in Figure 1, the convolution can be formulated as follows.

\[
y = \sigma(\sum_{k \in H} h^k \ast W^k + c)
\]

(2)

where the * denotes convolutional operation, and \( h^k \) represent the \( k \)th local field. The convolutional operation can convolute the M*M matrix and N*N matrix into (M+N-1)*(M+M-1) matrix (full convolutional operations) and (M-N+1)*(M-M+1) matrix (full convolutional operations). The cost function to minimize the mean square error (MSE):

\[
E(\theta) = \frac{1}{2n} \sum_{i=1}^{n} (x_i - y_i)^2
\]

(3)

Thus, the gradient of the cost function regarding the parameters is shown as follows

\[
\frac{\partial E(\theta)}{\partial W^k} = x \ast \delta h^k + h^k \ast \delta y.
\]

(4)

Where the \( x, h^k \) and \( y \) represents the input local field, hidden field and output field.
2.3. The optimization goals

Many researchers try to achieve balance between fit of data and ‘complexity’ of the solution in the learning algorithms. The way this complexity is defined differently in different algorithm. It is usually defined as a prior probability or a regularizer. Since the choice of the additional term depends on the learning problem to be addressed, the right choice should be decided by prior knowledge or assumptions about the problem. There is a useful assumption: two points that are close in input space should have the same label. Although sometimes the learning problems have more complicated structure than what this assumption captures, this assumption is widely applicable and useful. Based on above assumption, we add a regularizer in RBM, i.e. to regularize the hidden layer. It is natural to assume that if two points sampling from same class are close in the intrinsic geometry of the probability distribution, then their conditional probability \( p(h \mid x) \) are similar. In this case, the probability distribution that generates the data is supported on a sub-manifold of the ambient space.

\[
L = \sum_{i=1}^{N} \log P(x, y; \theta)
\]  

2.4. The overall Training Algorithm

The training of convolutional autoencoder as shown in Figure is decomposed into three stages. Because training the convolutional layers and fully-connected in a layer-wise manner involve too many layers, which make the model difficult to converge, we decouple the training of convolution layers and fully-connected layers. The details are shown as follows.

Stage 1: auto encoding for convolutional layers. As shown in Figure 1(a), there are five groups, each with several convolution layers and a pooling layer. The green layers represent convolutional layers including convolution operations, ReLu operations, norm operations. The blue layers denote pooling layers, which can deploy average pooling, max pooling and so on. The yellow layers mean the up-sampling layer. The encoding process and decoding process of this model are to minimize the construction error.

Stage 2: auto encoding for fully-connected layers. As shown in Figure 1(b), the nodes within the same layer have no connections, and the nodes between two contiguous layers are fully connected. As we discuss above, the auto-encoder model will result in identity mapping. Instead, we can use denoising auto encoders, sparse coding or probability model (Restricted Boltzmann Machine). As we aimed to learning a binary hash code, we choose to stake RBMs to generate the binary or binary-like hash code.

Stage 3: supervised learning for the pair-wise images. As shown in Figure 1(c). This stage makes the similar image learn closer codes.

3. Evaluation

3.1. Dataset

1) The first dataset we used is MNIST. The MNIST dataset consist of 60000 training and 10000 test handwritten digit images. These handwritten digit images are 28*28 binary images. We randomly choose 2000 samples from training data as training samples and 1000 samples from test data as test samples. Some parameters of RBM model is set as follows. The learning rate is 0.01. The number of iterations is 100. The number of hidden units is 100. The number of training images is 60000. The number of test images is 10000.

2) The CIFAR-10 dataset consists of 60000 32x32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

3) NUS-WIDE is a real-world web image database from national university of Singapore. Here we
introduce a web image dataset created by Lab for Media Search in National University of Singapore. Software, tools and Framework: Caffe, MATLAB, TensorFlow were used for CAE modeling and hashing.

3.2. Comparison of retrieval performance (mAP)

We conducted comprehensive experiments by comparing our method to other existing schemes and summarized results in Table 1.

Table 1. Retrieval performance (mAP) of Hamming rankings for different number of bits on the three image datasets

| Method   | MNIST       | CIFAR10     | NUS-WIDE    |
|----------|-------------|-------------|-------------|
|          | 12bit  | 24bit | 32bit | 48bit | 12bit | 24bit | 32bit | 48bit | 12bit | 24bit | 32bit | 48bit |
| Our Method | 0.935  | 0.940  | 0.938  | 0.950  | 0.427  | 0.451  | 0.409  | 0.468  | 0.615  | 0.451  | 0.409  | 0.468  |
| KSH       | 0.872  | 0.891  | 0.897  | 0.900  | 0.303  | 0.337  | 0.346  | 0.356  | 0.556  | 0.572  | 0.581  | 0.588  |
| ITQ-CCA   | 0.659  | 0.694  | 0.714  | 0.726  | 0.264  | 0.282  | 0.288  | 0.295  | 0.435  | 0.435  | 0.435  | 0.435  |
| MLH       | 0.472  | 0.666  | 0.652  | 0.654  | 0.182  | 0.195  | 0.207  | 0.211  | 0.500  | 0.514  | 0.520  | 0.522  |
| BRE       | 0.515  | 0.593  | 0.613  | 0.634  | 0.159  | 0.181  | 0.193  | 0.196  | 0.485  | 0.525  | 0.530  | 0.544  |
| SH        | 0.265  | 0.267  | 0.259  | 0.250  | 0.131  | 0.135  | 0.133  | 0.130  | 0.433  | 0.426  | 0.426  | 0.423  |
| ITQ       | 0.388  | 0.436  | 0.422  | 0.429  | 0.162  | 0.169  | 0.172  | 0.175  | 0.452  | 0.468  | 0.472  | 0.477  |
| LSH       | 0.187  | 0.209  | 0.235  | 0.243  | 0.121  | 0.126  | 0.120  | 0.120  | 0.403  | 0.421  | 0.426  | 0.441  |

3.3. Comparison with the State-of-the-art

We will use the following metrics to compare our model with other models. The metrics are shown as follows:

- mean Average Precision (mAP) for different code lengths.
- precision curves with 48 bits w.r.t. different number of top returned samples.

In order to compare our results with the state of art techniques, we compute the precision with respect to the number of images returned for 48bit codes as shown in Figure 2. We compare our retrieval performance with KSH, ITQ-CCA, MLH, BRE, SH, ITQ, LSH. Results show that our model shows substantial improvement under stage (1+2+3).

Figure 2: Retrieval Performance: Precision curves vs number of top returned images for the three Datasets compared with state of art techniques.

4. Conclusion

For content-based image retrieval, a good presentation is crucial. Nowadays, as deep learning models can be used to generate an excellent presentation, it has been extensively investigated and widely used in research systems and commercial production systems. However, the deep representation (deep feature) is still too large. For example, the 7th layer in classical AlexCNN is 4096. Compared with directly using deep representation, binary code can reduce significant storage overhead. Meanwhile, the bit-wise operations for binary code can dramatically fasten the computation. There exist some schemes used to
convert the deep feature to binary code, but all of them directly applied the last layer of fully connected layers, which represents global feature and discriminating features. In this paper, we use the generative model to transform the local feature to binary code using autoencoders. We proposed an architecture based on deep convolutional autoencoders to learn compact binary hash codes which reduces computation time. The convolutional layers help us to hierarchically learn features with preserving the semantic similarity. Empirical results show that on standard image retrieval datasets our model outperforms state-of-art unsupervised and supervised hashing techniques.

Acknowledgment
This project is supported in part by the US National Science Foundation Grant CCF-1337244, 1527249 and 1717388, and US Army/DURIP program W911NF-17-1-0208.

References
[1] XIA R., PAN Y., LAI H., LIU C., YAN S.: Supervised hashing for image retrieval via image representation learning. In AAAI (2014), vol. 1, p. 2.
[2] LAI H., PAN Y., LIU Y., YAN S.: Simultaneous feature learning and hash coding with deep neural networks. arXiv preprint arXiv:1504.03410 (2015).
[3] YANG H.-F., LIN K., CHEN C.-S.: Supervised learning of semantics-preserving hashing via deep neural networks for large-scale image search. arXiv preprint arXiv:1507.00101 (2015).
[4] LIU H., WANG R., SHAN S., CHEN X.: Deep supervised hashing for fast image retrieval. In Proceedings of the IEEE conference on computer vision and pattern recognition (2016), pp. 2064–2072.
[5] LIN K., YANG H.-F., HSIAO J.-H., CHEN C.-S.: Deep learning of binary hash codes for fast image retrieval. In Computer Vision and Pattern Recognition Workshops (CVPRW), 2015 IEEE Conference on (2015), IEEE, pp. 27–35.
[6] LI W.-J., WANG S., KANG W.-C.: Feature learning based deep supervised hashing with pairwise labels. arXiv preprint arXiv:1511.03855 (2015).
[7] KRIZHEVSKY A., HINTON G. E.: Using very deep autoencoders for content-based image retrieval. In ESANN (2011).
[8] WANG W., OOI B. C., YANG X., ZHANG D., ZHUANG Y.: BENGIO Y., ET AL.: Learning deep architectures for ai. Foundations and trends R in Machine Learning 2, 1 (2009), 1–127.