UNSUPERVISED DEEP HASHING WITH STACKED CONVOLUTIONAL AUTOENCODERS

Sovann En, Bruno Crémilleux, Frédéric Jurie

Normandie Univ, UNICAEN, ENSICAEN, CNRS — UMR GREYC

ABSTRACT

Learning-based image hashing consists in turning high-dimensional image features into compact binary codes, while preserving their semantic similarity (*i.e.*, if two images are close in terms of content, their codes should be close as well). In this context, many existing hashing techniques rely on supervision for preserving these semantic properties. In this paper, we aim at learning such binary codes by exploiting the underlying structure of unlabeled data, using deep learning. The proposed deep network is based on a stacked convolutional autoencoder which hierarchically maps input images into a low-dimensional space. A binary relaxation constraint applied to the middle layer of the network – the one containing the code – makes the codes sparse and binary. To demonstrate the competitiveness of the proposed architecture, we evaluate the so produced hash codes on image retrieval and image classification tasks on the MNIST dataset, and compare its performance with state-of-the-art approaches.

Index Terms— learning based hashing, unsupervised learning, convolutional autoencoder

1. INTRODUCTION

This paper addresses the problem of unsupervised feature learning, with the motivation of producing compact binary hash codes that can be used for indexing images. Such a representation is essential for many computer vision problems. For instance, in image retrieval, such representations enable efficient computations and low storage when dealing with large datasets [1, 2, 3, 4]. For image classification, it offers a higher level of semantic information compared to engineered feature extraction techniques [5, 6, 7].

Broadly speaking, there are two main ways to design hashing functions [8]: locality sensitive hashing (LSH) and learning to hash. LSH consists in combining data-independent hashing functions, while learning to hash relies on a supervised learning stage for producing hashing functions adapted to the input data, for a specific task. The former approach generally does not scale well, making it not suitable for large scale nearest neighbor search [9]. In addition, because of the locality preserving structure, LSH may not be able to preserve the semantic similarity.

Learning to hash techniques [3, 10, 11] have been proposed to overcome such drawbacks, relying on data annotations to train hashing functions which can map semantically similar examples to similar binary codes. Although they can help to address the aforementioned shortcomings, such approaches heavily rely on the availability of training data. This is the reason why a new trend in hashing is to consider semi-supervised or even unsupervised learning strategies [1, 6, 12, 13]. We build on this line of works.

Autoencoders are good candidates for unsupervised learning, due to their ability to learn data manifolds from unlabeled data (see *e.g.*, [14]). These networks are mostly based on regular autoencoders, made of fully connected layers, while recent state-of-the-art results are now also considering the use of *convolutional* autoencoders [15]. Inspired by the recent works of [2, 16], this paper takes advantage of convolutional deep autoencoders to design a novel and efficient way to design deep-hashing networks. While the objective of these aforementioned works is to learn hierarchical representations, our work is to focus on the production of compact binary codes.

In this paper, we learn to represent images by compact and discriminant binary codes, through the use of stacked convolutional autoencoders, relying on their ability to learn meaningful structure without the need of labeled data [6]. The layers are stacked to form a network which maps an input image hierarchically into a latent space, through a highly non-linear function (encoder part), and reconstruct hierarchically the original data (decoder part). Regular stacked convolutional autoencoders does not produce binary codes. We force the central layer, the one containing the code, to contain only binary values by introducing a regularization term in the loss function. This is one of the key contributions of this paper.

The capability of our network to produce semantic preserving codes is validated on the MNIST dataset on an image retrieval and image classification tasks. Experimental results illustrate the competitiveness of our network compared to state-of-the-art approaches.

The rest of the paper is as follows: Section 2 reviews the related works, Section 3 presents the method and the architecture, Section 4 gives experimental validation on image retrieval and classification while Section 5 concludes the paper.

2. RELATED WORK

Pioneer works in hashing [17] are based on the idea of Locality Sensitive Hashing (LSH), one of the most popular choices [9]. Variants of LSH such as the multi-probe LSH [18] or the non-linear hashing projection of [19] have also been investigated. Such approaches usually use hand-crafted features which does not provide an optimal solution for hashing.

Different from LSH, which relies on a random projection, learning based hashing takes advantages of modern learning algorithms to find an adapted hashing function on a given dataset. The hash functions can be developed in three different ways: supervised, semi-supervised and unsupervised learning. They differ from one to another in terms of the loss function and the constraints used. In supervised learning, the most important constraint is the preservation of semantic information. This is usually achieved by forcing the distance of the hash codes of similar (resp. dissimilar) data to be smaller (resp. bigger) than a margin [2, 4, 10, 11, 20, 21]. Semi-supervised hashing uses both labeled and unlabeled date. For instance, [3, 22] designed a loss function to minimize the distance between examples of the same class and maximize the variance between unlabeled data. In [12], in addition to pairwise similarity losses, the authors model the neighborhood structure of the labeled and unlabeled data. This structure acts as a regularizer so that the neighborhood of the data will be preserved in the hashing space. Unsupervised learning does not exploit any high level semantic information. For instance, [23] tries to minimize the quantization errors produced by the binarization of the features. [13, 23] design their hashing function based on the idea of preserving the neighborhood of the data in the Euclidean space.

Recent breakthrough in deep learning gradually attracted lots of researchers to propose new hashing techniques. [1] employs classical autoencoder to exploit the underlying structure of the data in an unsupervised manner followed by a RBM layer with constraints to reduce the dimension of the hamming space. [24] designed a new hashing technique based on variational autoencoders. Variational autoencoders assume the input data can be generated via a random probability distribution, which will be learned by the network.

Closely related to hashing techniques, [2] proposes a stacked what-where autoencoder based on convolutional autoencoders in which the necessity of switches (what-where) in the pooling/unpooling layers is highlighted. Another closely related work is the one of [16]. The authors utilize convolutional autoencoders but with an aggressive sparsity constraints. On each feature map, only the nb maximal activation is kept while the rest is put into lifetime sparsity (zero value). These closely related works, however, are designed for automatic features learning while our objective is to produce compact binary codes for hashing purpose.

3. MODEL ARCHITECTURE

Our model is based on a stacked convolutional autoencoder mapping input images into a compact latent space, through an encoder network, and reconstructing the original image through a decoder network. The network comprises of 3 convolutional autoencoders and one classical autoencoder. Each convolutional autoencoder can be summarized as: $conv - relu - batch\_normalization - pooling$ (encoder) and $unpooling - conv - relu$. The classical autoencoder is based on fully connected layer and can be compactly described as: fcxx-fcnb-fcxx. Figure 1 illustrates the whole architecture.

3.1. Convolutional Encoder-Decoder

More formally, let us assume that our network has L convolutional layers. The hierarchy of outputs given by the encoder is denoted as $x_i \in R^{W_i \times H_i \times C_i}$, for the $i^{th} - level$, where $W \times H$ is the size of the image and C the number of channels. On the other hand, the reconstructions given by the decoder are denoted as $\tilde{x_i} \in R^{W_i \times H_i \times C_i}$ for the $i^{th} - level$. x_0 is the input image and $\tilde{x_0}$ is the output of the network. The activation of the k^{th} layer of the encoder is defined by $x_k = \max\_\text{pool}(\sigma(x_{k-1} * W + b))$, while the activation of the k^{th} layer of the decoder is given by : $\tilde{x}_{k-1} = \text{unpool}(\sigma(\tilde{x}_k * \tilde{W} + \tilde{b}))$, where W/$\tilde{W}$ are the weights of the convolution/deconvolution layer, and b/$\tilde{b}$ the biases. $*$ denotes the convolutional operation. σ is the activation functions which are in our case rectified linear units except for the hashing layer in the middle of the network, and $\max\_\text{pool}$ is the max pooling operator (with a stride of 2 pixels in our case).

The parameters of the networks are learned with SGD, following a standard autoencoder optimization procedure where the loss function is the mean square error between input and reconstruction:

$$\underset{W,b}{\text{minimize}} \quad \ell_1 = \frac{1}{N} \sum_{j=1}^{N} \|\tilde{x}_0^j - x_0^j\|_2^2 \tag{1}$$

with N the amount of training samples. For simplicity, we drop all the subscripts of the weights W and biases b.

3.2. Fully connected central layer

Between the encoder and the decoder, we introduce a regular autoencoder made of 3 fully connected layers to further reduce the dimension of the features. The hidden layer of this autoencoder constitutes the code used to represent the images. It should be noticed that we also experimented with fully convolutional encoder-decoder network, without noticing much difference in terms of performance networks with a fully connected autoencoders. In practice, the fully convolutional network is faster to train and easier to optimize.

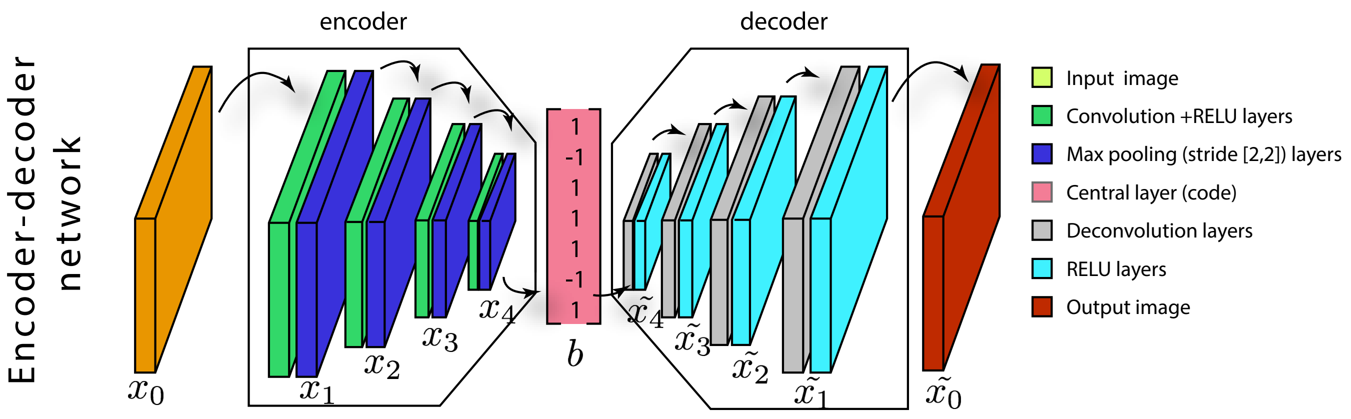

Fig. 1. Illustration of our network architecture. The left, middle and right blocks corresponds to the encoder, the fully connected layers and the decoder part. Each block in the encoder part consists of a convolutional, $relu$, batch normalization and a pooling layer. The fully connected layers, in the middle, learn to map the output of the encoder into a low-dimensional and binary spaces. The first and last FC layer are also equipped with $relu$. The decoder network is responsible to output the feature maps and gradually transform it into its original dimensions followed by 1x1 convolutional layer with sigmoid activation.

As for the fully convolutional encoder-decoder network, the optimization of the fully connected central autoencoder is based on the mean square error as given in Equation (1).

3.3. Producing sparse binary codes

While our goal is to learn a mapping function transforming input images into compact binary codes, the codes produced by the network given above are vectors of real values. In this section, we show how to introduce regularization terms making these codes binary, sparse and well distributed.

The binarization of the codes cannot be simply obtained by thresholding the real values of the codes. Such a thresholding would introduce a non-differentiable operation in the network, and standard back-propagation techniques would not be usable. This is why we opted for introducing a binary relaxation term [20]. This binary relaxation term aims at forcing the activation of the middle layer to values as close as possible to binary values $[-1, 1]$.

Let $b^j = (b_1^j, b_2^j, \ldots, b_d^j)^T$ be the activation vector of the hashing layer, for image j, d being the dimension of the code. The binary relaxation is done using:

$$\underset{W,b}{\text{minimize}} \quad \ell_2 = \frac{1}{N}\sum_{j=1}^{N}\sum_{i=1}^{d} \||b_i^j| - 1\| \tag{2}$$

In order to minimize Eq. (2), the activation b_i is pushed toward -1 or 1. The total loss function is then obtained by computing the weighted sum of $\ell 1$ end ℓ_2:

$$\underset{W,b}{\text{minimize}} \quad \ell = \frac{1}{N}\sum_{j=1}^{N} \|\tilde{x_0} - x_0\|_2^2 + \frac{\alpha}{N}\sum_{j=1}^{N}\sum_{i=1}^{d} \||b_i^j| - 1\| \tag{3}$$

Interestingly, autoencoders allow to represent the data in low-dimensional spaces. If the data can be reconstructed well from such low dimensional spaces, it means that these latent spaces contain enough high-level semantic information. In addition, making the codes sparse allows to index them more efficiently [1, 3, 16]. Assuming that we want only nb bits to be 1 and all the rest equal to -1, Eq. (3) can be rewritten as:

$$\underset{W,b}{\text{minimize}} \quad \ell = \frac{1}{N}\sum_{n=1}^{N} \|\tilde{x_n} - x_n\|_2^2 +$$
$$\frac{\alpha}{N}\sum_{j=1}^{N}\left(\sum_{i\in \mathrm{imax^j}} \|b_i^j - 1\| + \sum_{i\in \mathrm{imin^j}} \|b_i^j + 1\|\right)$$

where $\mathrm{imax^j}$ (resp. $\mathrm{imin^j}$) are the index of the first nb (resp. last $d - nb$) maximum values of b^j.

Theoretically, the nb bits should be distributed uniformly in the hashing layer. However, in practice, we notice that the network tends to focus on the same bits to encode all images. This makes our network having similar hash codes when adding the binarization term in the loss. Following [23], and to produce more balanced bits, we maximize the variance of each bit and force the bits to be pairwise uncorrelated. This can be done by setting the correlation matrix to be as close as possible to the identity matrix:

$$\frac{1}{N} b^T . b = I \tag{4}$$

It is worth mentioning that all the regularizations mentioned earlier are only applied to the hashing layer. However, as noted in [6, 16], without any regularization or denoising, autoencoder can barely learn any useful information. This is the reason why the $relu$ activation and the pooling layer are

Table 1. Retrieval mAP@1000 as a function of the number of bits of the hash code, on the MNIST dataset. XXX-VGGF denotes the hashing methods built on the top of features produced by the pre-trained VGG-F network. Our method produces its best overall performance with 32 filters in the convolutional layers.

Methods	12 bits	24 bits	32 bits	48 bits
ITQ-VGGF	0.407	0.478	0.487	0.506
SH-VGGF	0.301	0.304	0.296	0.287
LSH-VGGF	0.176	0.191	0.220	0.305
ITQ [23]	0.404	0.442	0.447	0.460
SH [21]	0.270	0.278	0.260	0.254
LSH [17]	0.162	0.236	0.222	0.276
Our method	**0.552**	**0.540**	**0.521**	**0.518**

used in the network. To some extent, these two layers can act as a good regularization to perturb the network not to learn to memorize all the information. In addition, the $relu$ activation also serves as a powerful non-linear model for the network to uncover the underlying structure in the data manifold.

4. EXPERIMENTATIONS

Hereafter, we describe the dataset used in our experimentations, give some implementation details and finally present our results and comparisons with state-of-the-art.

Datasets: The MNIST dataset consists of 70K 28×28 greyscale images of handwritten digits from 0 to 9. It consists of 60k training images and 10k testing images. Following [1, 11, 22], we randomly select 1k images (100 per class from the testing set) as our queries and keep the rest to learn the autoencoder. For image classification, we randomly selected a number of labeled data to train our classifier and test on the whole testing set. All the images are scaled so that their raw pixel intensities are in the range [0, 1].

Implementation: Each convolution layer is set to have $\{16, 32, 64\}$ filters with kernel size of 5x5. Each pooling and unpooling are set to reduce the dimension of the feature maps by 2 and vice versa (*i.e.* their stride is of [2,2]). Our model can be summarized as conv64-conv64-conv64-fc128-fc$X$-fc128-conv64-conv64-conv64 ($X \in \{12, 24, 36, 48\}$). The network is first trained in a layer-wise fashion before being fined tuned as a whole. Next, we apply the binary relaxation constraints by gradually increasing the α value from $1e^{-4}$ until the hash layer contains binary values (with a precision of $1e^{-3}$). Then, we fine tune again the network by gently increasing the β term (balancing bits regularizer) to make the hash codes well distributed.

Table 2. Classification error rate on the MNIST dataset using our network as a feature extractor, as a function of the numbers of labeled examples used to train the classifier. The dimension of the features in our case is of 12.

	Number of training examples			
	100	300	600	1000
SWWA [2]	0.118	0.058	0.042	0.025
CONV [5]	-	-	-	0.076
Our method	0.12	0.06	0.05	0.05

4.1. Image Retrieval Task

Table 1 shows the retrieval mAP@1000 obtained with our method, on the MNIST dataset, and compares it with state of the art methods. The mAP@1000 refers to the calculation of mAP on the 1000 first returned results for each query [1, 11]. The results are taken from the most recent works available on the MNIST dataset [12]. The proposed approach outperforms all the state-of-the-art of unsupervised method for hashing. The significant gain in performance fully justifies the competitiveness of our method compared to other recent unsupervised learning methods.

4.2. Image Classification Task

This section evaluates how our compact representation performs on classification tasks. Our model acts as a feature extractor, and we employ a simple k-NN classifier to show the robustness and discriminative power of our features.

Table 2 shows the classification errors obtained with our method, on the MNIST dataset, and compares it with state-of-the-art approaches. It should be noticed that the compared methods are trained end-to-end using labeled data, while our method act as an *unsupervised* feature learning method followed by a k-NN. Despite the fact that we do not use any annotations when learning the encoders, the performance is better or comparable to the state-of-the-art supervised techniques. This shows that our model successfully extract meaningful information from unlabeled data.

5. CONCLUSIONS

We proposed a novel architecture based on deep convolutional autoencoders to learn compact binary hash codes. The convolutional layers help us to hierarchically learn an embedding containing high-level semantic information. To turn the central layer into binary codes, a constraint based on binary relaxation is integrated together with a bit balancing regularizer. Experimental results demonstrate the competitiveness of our approach over state-of-the-art methods in image retrieval. Image classification experiments show that our model can also act as a good unsupervised feature learning, comparable to end-to-end networks without requiring any supervision.

6. REFERENCES

[1] Zhaoqiang Xia, Xiaoyi Feng, Jinye Peng, and Abdenour Hadid, "Unsupervised deep hashing for large-scale visual search," *arXiv preprint arXiv:1602.00206*, 2016.

[2] Junbo Jake Zhao, Michaël Mathieu, Ross Goroshin, and Yann LeCun, "Stacked what-where auto-encoders," *CoRR*, 2015.

[3] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang, "Semi-supervised hashing for scalable image retrieval," in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2010, pp. 3424–3431.

[4] Kevin Lin, Huei-Fang Yang, Jen-Hao Hsiao, and Chu-Song Chen, "Deep learning of binary hash codes for fast image retrieval," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 27–35.

[5] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *International Conference on Artificial Neural Networks*, 2011, pp. 52–59.

[6] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11.

[7] Diederik P Kingma and Max Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[8] Jingdong Wang, Ting Zhang, Jingkuan Song, Nicu Sebe, and Heng Tao Shen, "A survey on learning to hash," *arXiv preprint arXiv:1606.00185*, 2016.

[9] Jun Wang, Wei Liu, Sanjiv Kumar, and Shih-Fu Chang, "Learning to hash for indexing big data - a survey," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 34–57.

[10] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang, "Supervised hashing with kernels," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 2074–2081.

[11] Rongkai Xia, Yan Pan, Hanjiang Lai, Cong Liu, and Shuicheng Yan, "Supervised hashing for image retrieval via image representation learning.," in *AAAI*, 2014, vol. 1, p. 2.

[12] Jian Zhang, Yuxin Peng, and Junchao Zhang, "Ssdh: semi-supervised deep hashing for large scale image retrieval," *arXiv preprint arXiv:1607.08477*, 2016.

[13] Herve Jegou, Matthijs Douze, and Cordelia Schmid, "Product quantization for nearest neighbor search," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 1, pp. 117–128, 2011.

[14] Yuting Zhang, Kibok Lee, and Honglak Lee, "Augmenting supervised neural networks with unsupervised objectives for large-scale image classification," *arXiv*, 2016.

[15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[16] Alireza Makhzani and Brendan J Frey, "Winner-take-all autoencoders," in *Advances in Neural Information Processing Systems*, 2015, pp. 2791–2799.

[17] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al., "Similarity search in high dimensions via hashing," in *VLDB*, 1999, vol. 99, pp. 518–529.

[18] Wei Dong, Zhe Wang, William Josephson, Moses Charikar, and Kai Li, "Modeling lsh for performance tuning," in *Proceedings of the 17th ACM conference on Information and knowledge management*, 2008, pp. 669–678.

[19] Anirban Dasgupta, Ravi Kumar, and Tamás Sarlós, "Fast locality-sensitive hashing," in *ACM International conference on Knowledge discovery and data mining*, 2011, pp. 1073–1081.

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

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

[22] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang, "Semi-supervised hashing for large-scale search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 12, pp. 2393–2406, 2012.

[23] 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.

[24] Nat Dilokthanakul, Pedro AM Mediano, Marta Garnelo, Matthew CH Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan, "Deep unsupervised clustering with gaussian mixture variational autoencoders," *arXiv preprint arXiv:1611.02648*, 2016.