DEEP NETWORK-BASED IMAGE CODING FOR SIMULTANEOUS COMPRESSION AND RETRIEVAL

Qingyu Zhang, Dong Liu, Houqiang Li*

CAS Key Laboratory of Technology in Geo-Spatial Information Processing and Application System,
University of Science and Technology of China, Hefei 230027, China
zhqy@mail.ustc.edu.cn,{dongeliu,lihq}@ustc.edu.cn

ABSTRACT

Images on the Internet are usually in the form of compressed bitstream to save storage. To fulfill content-based image retrieval (CBIR), image features are also required to be stored in binary form. Can the bitstream of images and image features be unified and further condensed? Is it possible that the same binary code serves for compression and retrieval simultaneously? To address this problem, we make preliminary studies on a deep network-based image coding scheme in this paper. We first train a deep network for compressing images into bitstream, and then train another deep network for extracting image features as binary vector. We then combine the above two networks, and finetune the combined network using triplets of images for the task of CBIR. Our experimental results show that the proposed scheme achieves a compression ratio of 5.3 for 32×32 thumbnails, outperforms JPEG at similar compression ratios, and the resulting code is directly available for CBIR. Our work indicates a promising direction of simultaneous image compression and retrieval.

Index Terms— Content-based image retrieval (CBIR), Deep network, Image coding, Image compression.

1. INTRODUCTION

The amount of images on the Internet is keeping growing very fast. On the one hand, it calls for more efficient image compression methods to save the cost on storage and transmission of images. On the other hand, it urges the need of effective image retrieval methods to find out images of interest. Especially, content-based image retrieval (CBIR), i.e. retrieving images with similar content, can facilitate many applications such as e-commerce and surveillance. Therefore, both image compression and retrieval have been studied very intensively in the literature.

* Corresponding author. This work was supported by the National Program on Key Basic Research Projects (973 Program) under Grant 2015CB351803, by the Natural Science Foundation of China (NSFC) under Grants 61390512, 61331017, and 61325009, and by the Fundamental Research Funds for the Central Universities under Grant WK3490000001.

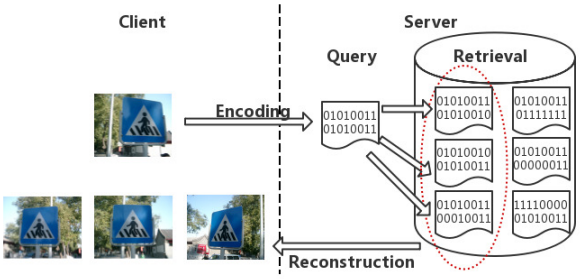

Fig. 1. Flowchart of our conceptually designed content-based image retrieval (CBIR) system.

Traditional image compression methods, such as the to date most commonly used JPEG [1], are designed to encode images into bitstream with the target to improve compression ratio. The coded bitstream is supposed to be used for only reconstructing images. Therefore, it is not easy to perform CBIR directly based on the coded bitstream of images, though several researches have been done to investigate this possibility [2]. Moreover, traditional CBIR methods often extract distinctive features from images and compare the image features to evaluate the similarity between images. Many different kinds of image features have been studied [3]. For a practical CBIR system, the storage/transmission cost of image features is probably a critical challenge, thus the compactness of image features is always pursued. Recently, compact descriptor for visual search (CDVS) has emerged as a competitive technique of encoding image features for CBIR, and has been a MPEG standard [4]. However, such compact descriptor is designed solely for CBIR but not for reconstructing images, though there are several researches on utilizing image features such as the well-known scale-invariant feature transform (SIFT) as *partial* information for reconstructing images [5].

Recently, deep learning has achieved remarkable progress in almost every area of image processing. For CBIR, more and more methods replace the traditional hand-crafted image features with the deep features, i.e. features extracted as output of a trained deep network [6]. Moreover, researches under the name of *deep hashing* have proposed to convert images into bit vectors, by means of deep networks, with the

target that the similarity between images is equivalent to the Hamming distance between their bit vectors [7–9]. For image compression, deep learning based methods are not well studied yet, while Toderici *et al.* have presented promising preliminary results [10, 11].

In this paper, we are interested in the question whether image compression and retrieval can be performed simultaneously. To be specific, our conceptually designed CBIR system is depicted in Fig. 1, with the objective that images are coded only once, and the coded bitstream can be used not only to reconstruct image, but also to be directly compared to search for similar images. To the best of our knowledge, such a CBIR system has not been studied before. However, the advantages of this system in practice are obvious. First, images and image features are compressed together to reduce the size of bitstream. Second, comparing bitstream is highly efficient for image retrieval.

Deep learning seems providing an approach to realize our conceptually designed CBIR system. Especially, the researches on deep hashing [7–9] and deep network-based image compression [10, 11] have demonstrated encouraging results. Following this line, we make preliminary studies on a deep network-based image coding scheme. Please note that image *coding* is used in the broad sense in this paper, i.e. converting images into bitstream. We adopt a step by step strategy to obtain a trained network for image coding. First, we train a deep network for compressing images into bitstream as well as reconstructing images, like an auto-encoder [12]. Second, we train another deep network for extracting image features as bit vector, driven by an image classification task. Third, the image encoder and feature extractor networks are combined and finetuned by means of triplet-wise training for the CBIR task. Our experimental results demonstrate the usability of the proposed image coding scheme. It achieves better reconstruction quality than JPEG at similar compression ratios, and the resulting code is verified to perform well in CBIR.

The remainder of this paper is organized as follows. Section 2 presents the strategy and design for training a deep network for image coding. Section 3 presents the details of implementation and experimental results. Section 4 concludes this paper.

2. APPROACH

Our purpose is to train a deep network for image coding so that the resulting code can be used both to reconstruct image and to search for similar images. How to train such a deep network seems not being reported before. As pre-training and finetuning have been extensively adopted in the deep learning literature, we propose to pre-train two separate networks that serve for compressing images and extracting image features, respectively, and then combine and finetune. The details are presented in this section.

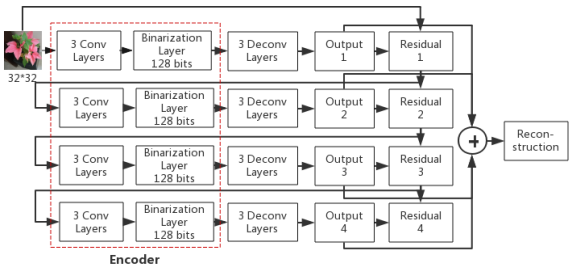

Fig. 2. The network structure to train encoder and decoder for compression.

2.1. The Encoder/Decoder Network for Compression

Since the proposal of auto-encoder [12], deep network is believed to benefit image compression conceptually, but a working encoder/decoder network that achieves state-of-the-art performance is not reported until recently [10, 11]. In this paper we follow the key idea proposed in [10] to design and train the encoder/decoder network.

As depicted in Fig. 2, the encoder/decoder network is split into several subnets (four subnets from top to bottom are shown in the figure). All the subnets have exactly the same structure but their parameters are individually trained. Each subnet has three convolutional layers followed by a binarization layer, as part of the encoder, and then three deconvolutional layers as part of the decoder. The subnets are trained sequentially. The first subnet accepts an image as input and is trained to approximate the image as accurately as possible. The parameters of the first subnet are fixed once training stops. Then, the residual between an original image and the corresponding output of the first subnet, i.e. *Residual 1* shown in Fig. 2, is taken as input to the second subnet which is trained to approximate that residual as possible. The parameters of the second subnet are then fixed once training stops. Then, the residual between Residual 1 and the corresponding output of the second subnet, i.e. *Residual 2*, is input to the third subnet, and so on. More subnets can be further trained one by one in the same manner. Therefore, the final encoder/decoder is composed by multiple subnets and the compression ratio can be adjusted by using different amounts of subnets. The reconstructed image would be sum of outputs produced by all the used subnets. In this paper we use four subnets.

2.2. Feature Extractor Network

One well-known advantage of deep learning is the automation of feature extraction from images, which eliminates the need of hand-crafted features. We revise the famous VGG network [13] to extract features as bit vector.

The original VGG network is depicted in Fig. 3 (top), having 16 convolutional layers followed by 3 full-connection (fc) layers. It is generally believed that the output at later

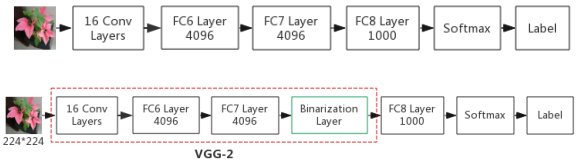

Fig. 3. Top: the original VGG network structure [13]. Bottom: the revised VGG network structure with a binarization layer inserted, the highlighted VGG-2 network is used as feature extractor.

layers of a deep network has higher level abstraction than that at earlier layers. Since the final fc layer, fc8, is indeed used for classification, we choose the output of fc7 as image features.

As the original network works on floating-point numbers that are costly to encode, we need to either reduce the amount of features or quantize these features, or both. We have empirically studied both solutions and find that quantization is better than reducing the amount. Accordingly, we insert a binarization layer after fc7, as shown in Fig. 3 (bottom), which quantizes the features to -1 or $+1$ (finally -1 is replaced by 0 in the bitstream).

Due to the insertion of binarization layer, we need to retrain the network properly. In our experiments, we use the parameters of the original VGG network to initialize. Then we *fix* the parameters of the convolutional layers but train the parameters of fc layers in Fig. 3 (bottom), driven by the ImageNet classification task [14]. After training, the convolutional layers, fc6, fc7, and the binarization layer, are composed together as *VGG-2* network that is our designed feature extractor.

2.3. Combined Network and Finetuning

For an input image, the encoder network and feature extractor network both output a bit vector. We can directly concatenate two bit vectors as the binary code, but this is sub-optimal as the two networks are separately trained. As we want the code to be directly comparable for CBIR, we propose to finetune the network for better retrieval performance.

Fig. 4 depicts the network structure for finetuning. We adopt the widely used triplet-wise training strategy [15] as it suits for CBIR. There are three copies of the combined network, each of which consists of a pre-trained encoder and a pre-trained VGG-2 network. The three copies of the combined network share parameters between them, and process three images known as anchor, positive, negative, respectively and individually. The triplet loss of a training sample (a, p, n) is defined as

$$L = \max(0, d(c_a, c_n) - d(c_a, c_p)) \quad (1)$$

where $d()$ stands for Hamming distance, c_a, c_p, c_n are the binary code of anchor, positive and negative, respectively. In the current implementation, we finetune the VGG-2 part to minimize the triplet loss but keep the encoder part unchanged, thus the decoder part for reconstructing image is also unchanged.

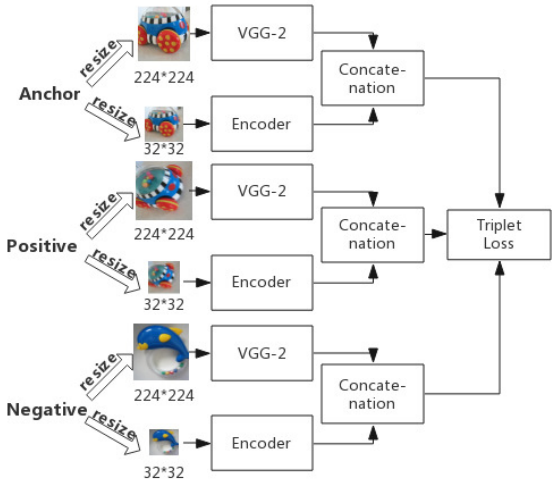

Fig. 4. The network structure to finetune the combined network (encoder as shown in Fig. 2 plus VGG-2 as shown in Fig. 3) using triplets of images for CBIR.

3. EXPERIMENTAL RESULTS

We perform experiments using the UKBench dataset [16] as it is widely adopted for CBIR researches. The dataset has 2550 groups of images, each group having 4 images, i.e. 10200 images in total. All the images have the same resolution at 640×480. For training encoder/decoder for compression, we randomly choose 9000 images from UKBench as training data, and the remaining 1200 images are used for testing the compression performance. Performance is evaluated by compression ratio and PSNR of reconstructed images. The VGG-2 network is pre-trained with an image classification dataset namely ILSVRC-2012[1]. For finetuning the combined network, we divide the UKBench dataset evenly into two parts, using one part (1275 groups, 5100 images) for training and the other part for testing the image retrieval performance. Note that during testing, each image in the test set is used as query image once, and the database consists of *all* the images of UKBench dataset. Thus we use the N-S score, i.e. 4 times the average precision of top-4 retrieval results, as the metric for evaluation. The ideal N-S score is 4 for UKBench dataset.

In our implementation, the encoder/decoder network takes 32×32 color images as input and compresses each image into 512 bits (64 bytes). The VGG-2 network takes 224×224 color images as input and the quantized features take 4096 bits (512 bytes) to encode. In total, each image is converted into bitstream of 576 bytes, the compression ratio for 32×32 images is constantly 5.3. But the bitstream

[1] http://www.image-net.org/challenges/LSVRC/2012/

Table 1. Comparison between JPEG and our trained encoder/decoder. Note that our combined network produces constant bit-rate for 32×32 images, i.e. 576 bytes, out of which the decoder actually uses only 64 bytes to reconstruct.

	JPEG	Ours
Average filesize (Bytes)	681.50	576 (64)
Average compression ratio	4.5	5.3 (48)
Average PSNR (dB)	23.16	23.39

Fig. 5. From top to bottom: original image, the reconstructed image with PSNR value by JPEG, and the reconstructed image with PSNR value by our trained encoder and decoder. Note that the compression ratios of JPEG and our encoder are comparable.

actually used for reconstructing image only takes 64 bytes, achieving a compression ratio of 48.

Table 1 and Fig. 5 present the results comparing our scheme and JPEG. For fair comparison, we resize the images in the test set to 32×32 and use the standard JPEG encoder to compress them. We adjust the quality parameter (QP) of JPEG to ensure that the reconstructed images have similar PSNR values, when QP is set to 8. It can be observed from Table 1 that our scheme outperforms JPEG in terms of both compression ratio and reconstructed quality (PSNR). Fig. 5 further demonstrates the improvement of visual quality of reconstructed images by our scheme than JPEG. At comparable compression ratios, our scheme better preserves the important structures in images.

Fig. 6 presents the results of image retrieval experiments, using our pre-trained VGG-2 network, directly combined network without finetuning, and our combined network with finetuning, respectively. Note that the combined network consists of the VGG-2 network as well as the encoder, and is finetuned with the training data, therefore, our combined network outperforms the other two networks consistently. The combined network achieves an N-S score of 3.46 on the testing data with 576 bytes per image and Hamming distance calculation between the binary codes. It then demonstrates the usability of our scheme for a practical CBIR system.

We also perform experiments to compare the methods of reducing the amount of features and quantizing the features for image feature compression. Table 2 shows the results

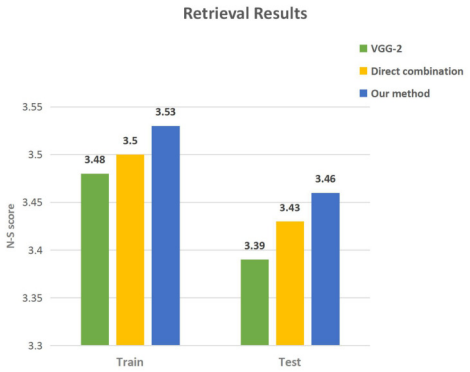

Fig. 6. The N-S score (4 times top-4 precision) results on the UKBench dataset, with VGG-2 (as shown in Fig. 3), directly combined network without finetuning, and our combined network with finetuning (as shown in Fig. 4) on the training and testing data, respectively.

Table 2. The N-S score (4 times top-4 precision) results on the UKBench dataset, using 128 floating-point numerical features or 4096 binary features.

	N-S Score
128 numerical features	2.05
4096 binary features	3.39

of using 128 features as floating-point numbers and using 4096 features as binary vector, both need 512 bytes to encode. Note that for floating-point numerical features we use the Euclidean distance to measure similarity between images, while for binary features we use the Hamming distance. Besides being computationally simpler, using binary features is shown to outperform using numerical features significantly. Therefore, we have adopted quantization of features in the VGG-2 network. In future work we plan to study the tradeoff between reducing the amount of features and increasing the precision (bit-depth) of features.

4. CONCLUSION

In this paper we present a deep network-based image coding scheme for simultaneous compression and retrieval, i.e. the produced bitstream can be used to reconstruct image as well as to be directly compared to search for similar images. The deep network is trained step by step: first the encoder/decoder network for compression, second the image feature extractor network, and third the combined network finetuned with triplet-wise training. Experimental results have verified the usability of the scheme as it indeed achieves compression ratio similar to JPEG and the resulting code is directly available for CBIR.

The experimental results presented in this paper are indeed preliminary. There are many ways to improve the deep network for better performance. Especially, it would be beneficial to train a network with joint reconstruction loss and similarity preserving (e.g. triplet-wise) loss.

5. REFERENCES

[1] Gregory K Wallace, "The JPEG still picture compression standard," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.

[2] David Edmundson and Gerald Schaefer, "An overview and evaluation of JPEG compressed domain retrieval techniques," in *Proceedings ELMAR*. IEEE, 2012, pp. 75–78.

[3] Ying Liu, Dengsheng Zhang, Guojun Lu, and Wei-Ying Ma, "A survey of content-based image retrieval with high-level semantics," *Pattern Recognition*, vol. 40, no. 1, pp. 262–282, 2007.

[4] Ling-Yu Duan, Jie Lin, Jie Chen, Tiejun Huang, and Wen Gao, "Compact descriptors for visual search," *IEEE Multimedia*, vol. 21, no. 3, pp. 30–40, 2014.

[5] Huanjing Yue, Xiaoyan Sun, Jingyu Yang, and Feng Wu, "Cloud-based image coding for mobile devices–Toward thousands to one compression," *IEEE Transactions on Multimedia*, vol. 15, no. 4, pp. 845–857, 2013.

[6] Ji Wan, Dayong Wang, Steven Chu Hong Hoi, Pengcheng Wu, Jianke Zhu, Yongdong Zhang, and Jintao Li, "Deep learning for content-based image retrieval: A comprehensive study," in *ACM Multimedia*. ACM, 2014, pp. 157–166.

[7] Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, and Jie Zhou, "Deep hashing for compact binary codes learning," in *CVPR*, 2015, pp. 2475–2483.

[8] Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen, "Deep supervised hashing for fast image retrieval," in *CVPR*, 2016, pp. 2064–2072.

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

[10] George Toderici, Sean M O'Malley, Sung Jin Hwang, Damien Vincent, David Minnen, Shumeet Baluja, Michele Covell, and Rahul Sukthankar, "Variable rate image compression with recurrent neural networks," *arXiv preprint arXiv:1511.06085*, 2015.

[11] George Toderici, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor, and Michele Covell, "Full resolution image compression with recurrent neural networks," *arXiv preprint arXiv:1608.05148*, 2016.

[12] Geoffrey E Hinton and Ruslan R Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[13] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[14] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei, "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[15] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman, "Deep face recognition," in *BMVC*, 2015, p. 6.

[16] David Nister and Henrik Stewenius, "Scalable recognition with a vocabulary tree," in *CVPR*, 2006, vol. 2, pp. 2161–2168.