CNN-Based DCT-Like Transform for Image Compression

Dong Liu(✉), Haichuan Ma, Zhiwei Xiong, and Feng Wu

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

Abstract. This paper presents a block transform for image compression, where the transform is inspired by discrete cosine transform (DCT) but achieved by training convolutional neural network (CNN) models. Specifically, we adopt the combination of convolution, nonlinear mapping, and linear transform to form a non-linear transform as well as a non-linear inverse transform. The transform, quantization, and inverse transform are jointly trained to achieve the overall rate-distortion optimization. For the training purpose, we propose to estimate the rate by the l_1-norm of the quantized coefficients. We also explore different combinations of linear/non-linear transform and inverse transform. Experimental results show that our proposed CNN-based transform achieves higher compression efficiency than fixed DCT, and also outperforms JPEG significantly at low bit rates.

Keywords: Convolutional neural network (CNN)
Discrete cosine transform (DCT) · Rate-distortion optimization
Transform

1 Introduction

Transform coding is a fundamental technique in modern image and video compression solutions. Almost all of the existing lossy compression schemes adopt transform, but in different forms. For example, JPEG adopts 8×8 discrete cosine transform (DCT) [1], JPEG2000 adopts several kinds of discrete wavelet transform [2], MPEG-4 AVC/H.264 adopts integer cosine transform at different block sizes like 4×4 and 8×8 [3], which is inherited and extended to more block sizes in HEVC [4]. The pursuit of higher compression efficiency continuously inspires the emergence of new transform tools, such as graph-based transform [5].

Traditionally, transform is derived based on the signal processing theory, which often assumes stationary signal to claim the optimum of the derived transform. However, the image/video signal is non-stationary and usually too complicated to be analyzed precisely. Then, how to derive transform that works better for image/video is a difficult problem that probably goes beyond the current signal processing theory.

K. Schoeffmann et al. (Eds.): MMM 2018, Part II, LNCS 10705, pp. 61–72, 2018.
https://doi.org/10.1007/978-3-319-73600-6_6

In recent years, deep learning receives more and more attention especially in the fields of computer vision and image processing. There have been some works studying deep learning-based image/video compression, which is demonstrated to be a promising direction [6–14].

A large portion of previous works, such as [6–11], are devoted to end-to-end training of deep network models for image compression. Indeed, these works were inspired by the paradigm of auto-encoder, which originates from the well-known work of Hinton and Salakhutdinov [15]. The original auto-encoder network consists of two parts that perform "encoding" and "decoding," respectively, and is pre-trained layer by layer and then fine-tuned end to end. In the original auto-encoder network, the dimension of "encoded" features is configured to be less than the dimension of input signal, so that the "encoding" part of the network performs non-linear dimensionality reduction; but the features are floating-point numbers and thus the network is not quite useful for practical compression. In the following works [6–11], a quantization layer is inserted into auto-encoder to convert the "encoded" features into bits or integers that are then collected into bit-stream. These works can be categorized into two approaches. The first approach works at block level, converts blocks of fixed size (32×32) into bits [6–8]. The second approach works at image level by converting the entire input image into integers [9–11]. The difference between these two approaches are similar to that between JPEG and JPEG2000.

Although the existing works [6–11] have demonstrated the great potential of using auto-encoder-like schemes for image compression, there are several limitations in these works. First, the networks using binary quantization [6–8] cannot provide arbitrary bitrate since the possible bitrates are predefined by the network structure. Second, the networks working at image level lack the flexibility of block-based image/video compression, i.e. enabling adaptive mode selection at block level [3,4]. Third, the new schemes are totally different from the existing standards (JPEG, JPEG2000, H.264, HEVC, etc.), so it is difficult to integrate them into the standards, or to reuse the existing techniques in the standards.

In this paper, we study a deep learning-based method for transform coding. We also follow the paradigm of auto-encoder to train an end-to-end convolutional neural network (CNN) model. But different from the previous works, our key idea is to find ways to improve the existing DCT rather than to develop a totally new tool. Thus, our trained transform can replace DCT so as to be seamlessly integrated into the block-based image/video compression standards, such as HEVC. Our network works at block level to embrace the flexibility of block-based schemes, and our network adopts multi-level quantization rather than binary quantization so as to enable arbitrary bitrate. In addition, we make the following new contributions compared to previous works. First, we investigate asymmetric auto-encoder schemes in order to reduce the computational complexity of encoding or decoding. Second, we propose to use the l_1-norm of quantized coefficients to estimate rate when training the network for rate-distortion optimization.

The remainder of this paper is organized as follows. Section 2 reviews the related works. Section 3 discusses the proposed method. Section 4 presents the experimental results, followed by conclusions in Sect. 5.

2 Related Works

Deep learning-based image/video compression has become an emerging topic and received much attention in the recent two years [6–14]. Most of the previous works are devoted to auto-encoder-like schemes for image compression. As mentioned above, these works can be categorized into two approaches, block-level and image-level.

For networks working at block level, Toderici *et al.* proposed a scheme based on recurrent neural network (RNN) models [6]. The network has a binary quantization layer inserted after the "encoding" part of auto-encoder, so the bitrate is predefined by the network structure. In order to achieve variable bitrates, either multiple auto-encoders are cascaded, or RNN is run for multiple iterations. Later on, they improved the scheme by further compressing the network output bits using a carefully designed RNN-based arithmetic coding method [7]. More recently, Johnston *et al.* further improved the scheme by enabling priming in RNN and allocating different bitrates to different blocks [8]. All these works adopt binary quantization, so the bitrate cannot be made arbitrary.

For networks working at image level, Balle *et al.* proposed a scheme based on fully convolutional network models [9]. The network adopts multi-level quantization (i.e. quantization to integers) rather than binary quantization, so different quantization steps naturally lead to arbitrarily variable bitrates. In addition, to train the network including quantization, they proposed a new loss function to approximate rate-distortion cost. Theis *et al.* [10] and Rippel and Bourdev [11] also proposed similar schemes but with different network structures and different ways to approximate rate-distortion cost. These schemes are not easy to be integrated into the existing standards like HEVC.

Besides the auto-encoder-like schemes, there are some other explorations of deep learning-based image/video compression. For example, Jiang *et al.* proposed a CNN-based framework for image compression, where an input image is down-scaled by a CNN, compressed, and then up-scaled by another CNN [12]; Baig and Torresani proposed a deep learning-based method for colorization, and adopted the trained network to compress the chrominance components of images [13]; Prakash *et al.* proposed an image compression method with CNN-based prediction of perceptual distortion [14]. Nonetheless, there are only a few researches of deep learning-based predictive coding and transform coding, which are worthy of further investigation.

3 The Proposed Method

3.1 Mathematical Formulations

We first define some symbols for describing block transform for image compression. An original image block $\mathbf{B} \in \mathbb{R}^{N \times N}$ is transformed into coefficients

$\mathbf{C} \in \mathbb{R}^{N \times N}$, and then quantized into integers $\mathbf{I} \in \mathbb{Z}^{N \times N}$. The quantized coefficients are inversely transformed to reconstruct the image block $\hat{\mathbf{B}} \in \mathbb{R}^{N \times N}$. There are three steps performing transform, quantization, and inverse transform, respectively:

$$\mathbf{C} = f_E(\mathbf{B}) \tag{1}$$

$$\mathbf{I} = f_Q(\mathbf{C}) \tag{2}$$

$$\hat{\mathbf{B}} = f_D(\mathbf{I}) \tag{3}$$

In this paper we study the case of $N = 32$, to be compared with the previous works [6–8]. Please note that the amount of coefficients is identical to the amount of pixels, i.e. no dimensionality reduction is performed, which is different from most of the previous works that follow [15].

Our aim is to derive a good transform for natural images via deep learning. That is, the transform, quantization, and inverse transform functions are integrated into a unified deep network; given a set of image blocks $\{\mathbf{B}_i\}$, the deep network can be trained end to end in order to optimally compress the blocks. We consider to optimize the joint rate-distortion cost, where distortion is measured by mean-squared-error (MSE):

$$D = \sum_i ||\mathbf{B}_i - \hat{\mathbf{B}}_i||_2^2 \tag{4}$$

There is a difficulty to measure the rate, which is dependent on the entropy of $\mathbf{I}_i$, but the entropy is not differentiable and thus cannot be put directly into the deep network for training. There have been several proposals to estimate the entropy with mathematical tractable functions [9–11], but these functions are computationally expensive. In this paper, we are motivated by the well-known rate-ρ relation in image/video coding, which states the rate is highly dependent on ρ, the amount of non-zero quantized coefficients [16]. Actually, ρ is the l_0-norm of the quantized coefficients, and l_0-norm minimization implies sparsity constraint [17], but l_0-norm is mathematically intractable. We further replace l_0-norm with l_1-norm since l_1-norm minimization also implies sparsity constraint but is much easier to solve [17]. Therefore, we propose to measure the l_1-norm of the quantized coefficients to estimate the rate:

$$R = \sum_i ||\mathbf{I}_i||_1 \tag{5}$$

Then, the joint rate-distortion cost is used as the total loss function to train the deep network,

$$\{f_E^*, f_Q^*, f_D^*\} = \arg \min_{f_E, f_Q, f_D} (D + \lambda R) \tag{6}$$

where λ is the Lagrangian multiplier. Note that for different λ values, the optimal network parameters can be different.

In this paper we focus on the transform and the inverse transform, so for simplicity we fix the quantization function to be rounding, i.e. $f_Q(\cdot) = [\cdot]$. Since the rounding function is not differentiable, as a workaround, we adopt the trick proposed in [6], i.e. $f_Q'(\cdot) \approx 1$.

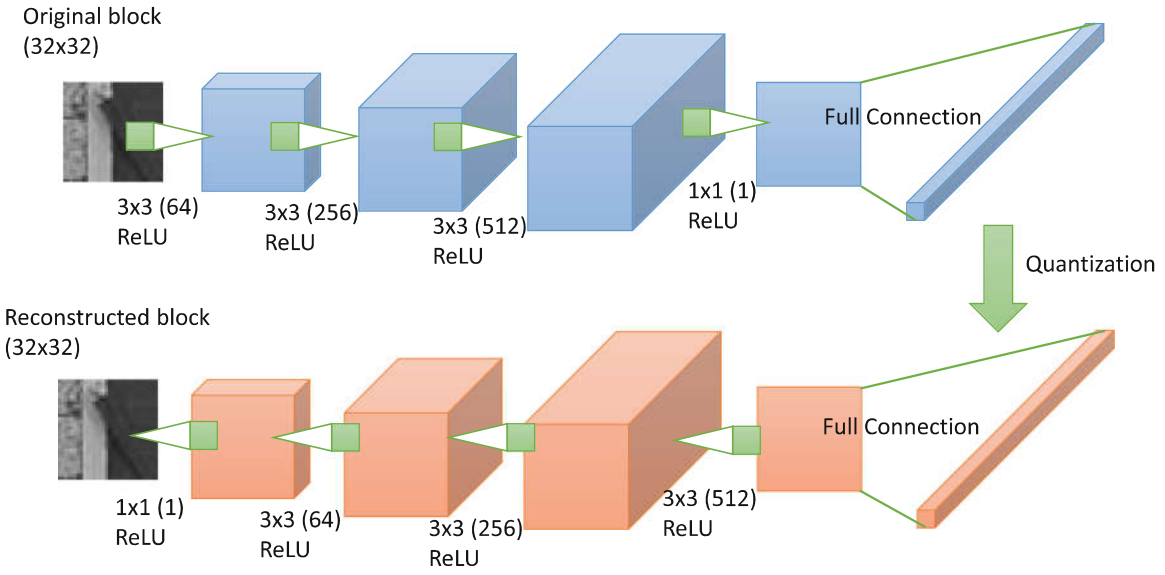


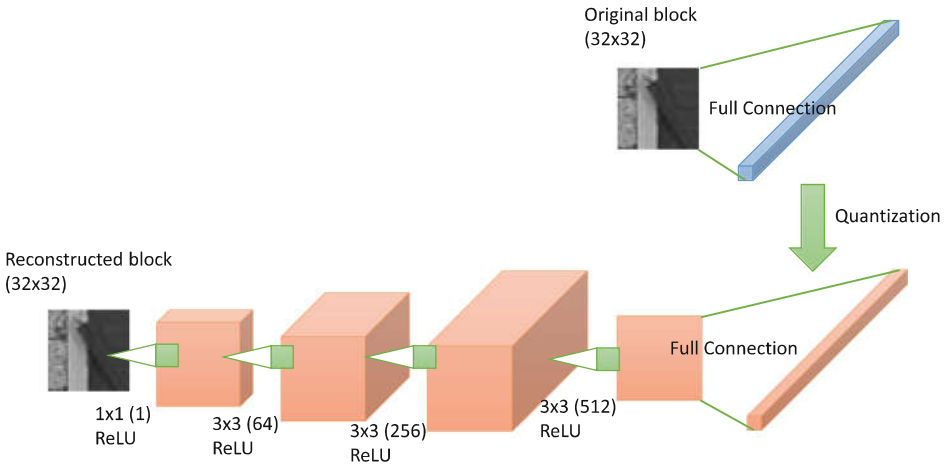
Fig. 1. The network to train a non-linear transform and a non-linear inverse transform, called symmetric network hereafter. For each convolutional layer, the shown numbers indicate kernel size (such as 3×3) and the amount of feature maps (such as 64). Each convolutional layer is followed by a non-linear mapping, which is rectified linear unit (ReLU) [18] in this paper (i.e. $f(x) = \max(0, x)$). The full-connection layers have 1024 inputs and 1024 outputs. The quantization layer performs rounding.

3.2 Non-linear Transform and Non-linear Inverse Transform

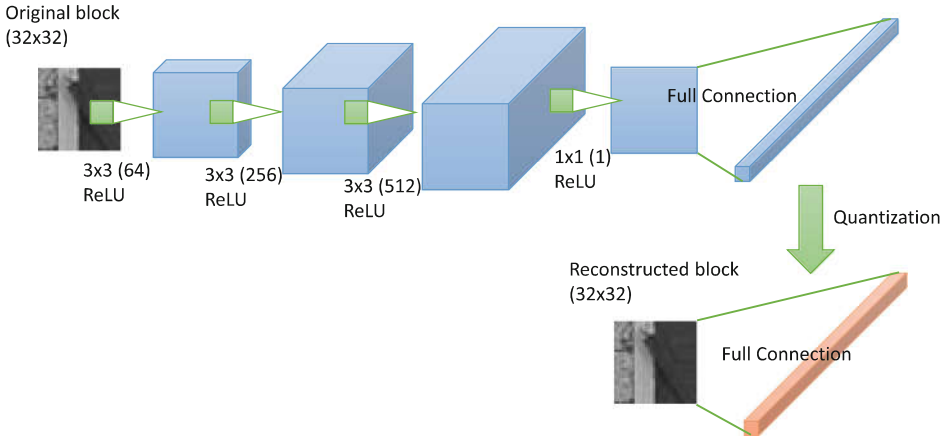
We now introduce the transform and the inverse transform realized by deep network. Firstly, we investigate the combination of a non-linear transform and a non-linear inverse transform, which is depicted in Fig. 1. The network can be interpreted as to perform sequentially: pre-processing on the original block (with four convolutional layers), linear transform (with full-connection layer), quantization, linear inverse transform (with full-connection layer), and post-processing to produce the reconstructed block (with four convolutional layers).

For this network, if we omit the pre- and post-processing parts, and initialize the full-connection layers as DCT and inverse DCT, respectively, then the entire network actually performs DCT, quantization, and inverse DCT. When adding the pre- and post-processing parts and training the entire network end to end, we expect it to outperform the original DCT for image compression, because the trained network is tuned according to the given training data to minimize the joint rate-distortion cost.

Compared our network with the previously designed auto-encoder-like schemes, there are several notable differences. First, there is no change of scale/resolution throughout our network, but down- and up-scaling are extensively used in [7,9]. Second, there is full-connection layers in our network, but previous works adopt merely convolutions [7,9]. We would like to remark that convolutions do not naturally remove the spatial redundancy among pixels, and thus the combination of convolutions and down-scaling is necessary like in JPEG2000. However in our network, we use the full-connection layer to remove the spatial redundancy, thus avoid down-scaling.



(a) Simplified encoder



(b) Simplified decoder

Fig. 2. Two asymmetric networks with simplified encoder and simplified decoder, respectively.

3.3 Linear and Non-linear Transform and Inverse Transform

We also investigate two different networks that are depicted in Fig. 2. Compared to the network in Fig. 1, the two networks in Fig. 2 either omit the pre-processing parts so that the original block is directly linearly transformed to coefficients (Fig. 2(a)), or omit the post-processing parts so that the quantized coefficients are directly linearly transformed to the reconstructed block (Fig. 2(b)). We call these two networks *asymmetric* networks and in contrast the network in Fig. 1 is called *symmetric*. To the best of our knowledge, we are the first to study asymmetric networks in the paradigm of auto-encoder. For practical compression, one obvious advantage of asymmetric auto-encoder is the reduction of computational complexity in either encoder or decoder. For example, the simplified encoder network (Fig. 2(a)) is suitable for usage when an image is uploaded from a mobile device to the cloud, while the simplified decoder network (Fig. 2(b)) is suitable when an image is downloaded from the cloud to a mobile device, where

the mobile device has limited computational resource but the cloud has much. Thus, we would like to observe their compression performance experimentally.

(a) Our simplified encoder, 0.272 bpp, 31.95 dB

(b) Toderici *et al.* , 0.250 bpp, 27.63 dB

(c) Fixed 32×32 DCT, 0.245 bpp, 30.46 dB

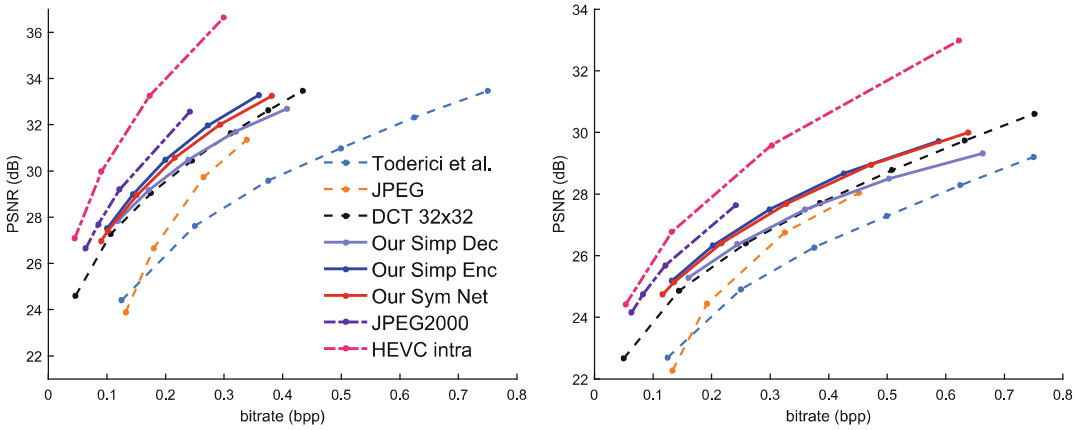
(d) JPEG, 0.265 bpp, 29.73 dB

Fig. 3. Reconstructed images for kodim07 at around 0.25 bpp.

4 Experimental Results

4.1 Training

We use the Uncompressed Colour Image Database (UCID) that consists of 1,338 natural images [19] to prepare the training data. The images are converted to grayscale and divided into 32×32 blocks, and pixel values are divided by 255 to fall into the range $[0, 1]$. No other processing step is involved. We use Caffe [20] and stochastic gradient descent (SGD) to train the networks depicted in Figs. 1 and 2. The full-connection layers are initialized by the weights of DCT and inverse DCT, respectively. We observed that such initialization helps to converge very fast. Each network is first trained by setting $\lambda = 0$, i.e. only considering distortion, until convergence. At this time, the reconstruction quality is usually very high (more than 45 dB), showing that the network is well trained. Next, for a given λ, we proceed to train the network until convergence. Thus, for different λ values the trained network parameters are different, which can



(a) kodim07 (b) kodim14

Fig. 4. Rate-distortion curves of different methods on two test images.

be used to provide variable bitrates for the same input. In this paper, the λ values are chosen from $\{10^{-4}, 8\times10^{-5}, 4\times10^{-5}, 2\times10^{-5}, 10^{-5}, 5\times10^{-6}\}$. As λ decreases, both reconstruction quality and bitrate increase, showing the loss function indeed plays the role of joint rate-distortion cost.

4.2 Image Compression with the Trained Networks

When using a trained network for compression, the network is split into encoder and decoder at the quantization (rounding) layer. An image to compress is divided into 32×32 blocks, with its pixel values divided by 255, then input into the encoder to achieve integer codes. The integer codes are fed into decoder to reconstruct blocks. To further compress the integer codes, note that our trained transform is improved upon DCT (and the full-connection layer is initialized by DCT), we assume the transform coefficients are corresponding to DCT coefficients, and thus we can reuse the entropy coding method designed for DCT coefficients. In experiments, we implement such an entropy coding method: for the first coefficient, corresponding to direct current (DC), it is predicted from the DC of the previous block, and the prediction residue is coded; for the other coefficients, corresponding to alternating current (AC), they are arranged in the zigzag scanning order, and then the run-length of 0's is coded followed by each non-zero coefficient; all the syntax elements are collected by a multi-level arithmetic encoder [21].

4.3 Compared Methods

We use the 24 images in the Kodak image dataset[1] to test the performance. The raw color images are converted to grayscale before compression. We compare the three networks presented in this paper, i.e. the symmetric network (shown

[1] http://r0k.us/graphics/kodak/.

in Fig. 1), the simplified encoder and the simplified decoder (shown in Fig. 2(a) and (b), respectively). Besides, we compare with fixed 32 × 32 DCT: for fair comparison, it uses the same entropy coding method designed by ourselves. We also compare with JPEG, JPEG2000, HEVC intra, and the method proposed by Toderici *et al.* [7]. For JPEG and JPEG2000 we use the implementation provided in MATLAB, for HEVC intra we use the HM codec[2], and for Toderici *et al.* we use the trained network provided by the authors[3].

4.4 Results

Figure 3 presents the reconstructed images using different methods for kodim07 at around 0.25 bpp. The results of fixed 32 × 32 DCT and JPEG suffer from severe ringing artifacts in the area of leaves and flowers. In our result, the ringing is greatly reduced. The result of Toderici *et al.* is more smooth in the area of leaves, but has strange texture in the area of window frame. Overall, our result has the best visual quality among them. Note that our result has the highest PSNR at similar bit rates.

Figure 4 depicts the rate-distortion curves of different methods for two representative test images, where distortion is measured by PSNR. Our method achieves better performance than fixed DCT in a wide range of bitrates. Both our method and fixed 32 × 32 DCT outperform JPEG significantly, especially at low bitrates. However, note that Toderici *et al.* performs worse than JPEG, and JPEG2000 and HEVC intra are the best performers among the compared methods. Thus, it is not trivial to outperform the existing highly optimized standards.

When comparing the three networks proposed in this paper, we find the symmetric network usually performs the best among the three, but the simplified encoder network also performs very well, and even surpasses the symmetric network on several test images (e.g. kodim07, shown in Fig. 4(a)). On the contrary, the simplified decoder network does not compete favorably with the other two, and is often worse than fixed DCT. Therefore, it seems critical to adopt nonlinear inverse transform to pursue higher compression efficiency. The simplified encoder network, though performing worse than the symmetric network, is still better than fixed DCT, and thus can be useful in scenarios where the encoding complexity is concerned more than compression efficiency.

Table 1 summarizes the BD-rate results of our symmetric network compared with different anchors including fixed 32 × 32 DCT, JPEG, and Toderici et al. When comparing with fixed 32×32 DCT, our method achieves BD-rate reduction for 22 out of 24 test images, except for kodim15 and kodim17. The average BD-rate reduction of 24 images is 8.83%. When comparing with JPEG and Toderici et al., our method achieves significant BD-rate reduction for all of the test images, with on average 38.03% and 56.66% BD-rate reduction, respectively.

[2] https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.15/.

[3] https://github.com/tensorflow/models/tree/master/compression. This network has no entropy coding since the authors do not provide.

Table 1. BD-rate results of our symmetric network compared with different anchors

	Ours vs. DCT (32×32)	Ours vs. JPEG	Ours vs. Toderici *et al.*
kodim01	−17.74%	−28.45%	−36.71%
kodim02	−3.15%	−56.38%	−79.21%
kodim03	−10.28%	−47.22%	−67.99%
kodim04	−2.86%	−50.96%	−65.11%
kodim05	−18.31%	−24.65%	−28.60%
kodim06	−13.29%	−35.05%	−59.77%
kodim07	−11.10%	−39.13%	−54.94%
kodim08	−11.63%	−24.42%	−36.11%
kodim09	−8.09%	−41.15%	−61.45%
kodim10	−5.46%	−42.18%	−61.39%
kodim11	−10.91%	−33.09%	−55.52%
kodim12	−8.84%	−43.60%	−69.24%
kodim13	−13.35%	−23.27%	−41.25%
kodim14	−15.91%	−30.20%	−46.09%
kodim15	*7.17%*	−37.82%	−60.90%
kodim16	−7.95%	−46.79%	−67.92%
kodim17	*11.88%*	−35.22%	−44.26%
kodim18	−15.79%	−34.46%	−48.31%
kodim19	−9.84%	−47.52%	−65.65%
kodim20	−3.26%	−35.89%	−62.65%
kodim21	−17.32%	−34.89%	−60.27%
kodim22	−12.52%	−39.39%	−57.85%
kodim23	−3.48%	−54.09%	−77.02%
kodim24	−10.00%	−26.80%	−51.71%
Average	−8.83%	−38.03%	−56.66%

Table 2 presents the BD-rate results of different methods compared with JPEG. It can be observed that our symmetric network performs better than the simplified encoder and the simplified decoder networks. However, JPEG2000 and HEVC intra perform even better. For comparison purpose, we include into Table 2 the BD-rate results reported in [8]. Please note that our results are achieved with grayscale images converted from the Kodak image dataset, and the results in [8] are achieved with the color images, so the BD-rate results are not directly comparable. All the results show that deep learning-based methods including ours and those in [8,10] are better than JPEG but still worse than JPEG2000 and HEVC intra. Thus, it may be not wise to replace the existing standards with an entirely new deep learning-based scheme. Finally, we would like to remark again that our method provides an improved transform than

Table 2. BD-rate results of different schemes compared with JPEG

Test set	Method	BD-rate
Kodak (grayscale)	Our symmetric network	−38.03%
	Our simplified encoder	−31.13%
	Our simplified decoder	−18.31%
	JPEG2000	−54.89%
	HEVC intra	−71.00%
Kodak (color)[a]	Toderici *et al.* [7]	*13.34%*
	Johonston *et al.* [8]	−27.14%
	Theis *et al.* [10]	−29.04%
	JPEG2000[b]	−38.28%
	HEVC intra[c]	−54.85%

[a]These results are quoted from Table 3 of [8]
[b]Using OpenJPEG codec (http://www.openjpeg.org/)
[c]Using BPG codec (https://bellard.org/bpg/) and YUV 420 format

fixed DCT, and thus our method can be seamlessly integrated into block-based schemes like HEVC. We will study the integration in the future.

5 Conclusion

This paper presents a block transform inspired by DCT and achieved by training convolutional neural network models. Our network consists of convolution, non-linear mapping and linear transform so as to provide non-linear transform and non-linear inverse transform. We propose to estimate the rate by the l_1-norm of the quantized coefficients, and thus the network can be trained end to end. We also study asymmetric structures of the network. Experimental results show that the trained transform achieves better compression performance than fixed DCT.

Our future work will proceed in three directions. First, we plan to investigate other network structures to achieve higher compression efficiency. Second, we will extend the transform to other block sizes like 16×16 and 8×8. Third, we will integrate the trained transform into the state-of-the-art video coding scheme such as HEVC.

Acknowledgment. This work was supported by the Natural Science Foundation of China (NSFC) under Grant 61772483, Grant 61390512, and Grant 61425026, and by the Fundamental Research Funds for the Central Universities under Grant WK3490000001.

References

1. Wallace, G.K.: The JPEG still picture compression standard. IEEE Trans. Consum. Electron. **38**(1), xviii–xxxiv (1992)

2. Christopoulos, C., Skodras, A., Ebrahimi, T.: The JPEG2000 still image coding system: an overview. IEEE Trans. Consum. Electron. **46**(4), 1103–1127 (2000)
3. Wiegand, T., Sullivan, G.J., Bjontegaard, G., Luthra, A.: Overview of the H.264/AVC video coding standard. IEEE Trans. Circ. Syst. Video Technol. **13**(7), 560–576 (2003)
4. Sullivan, G.J., Ohm, J., Han, W.J., Wiegand, T.: Overview of the high efficiency video coding (HEVC) standard. IEEE Trans. Circ. Syst. Video Technol. **22**(12), 1649–1668 (2012)
5. Hu, W., Cheung, G., Ortega, A., Au, O.C.: Multiresolution graph fourier transform for compression of piecewise smooth images. IEEE Trans. Image Process. **24**(1), 419–433 (2015)
6. Toderici, G., O'Malley, S.M., Hwang, S.J., Vincent, D., Minnen, D., Baluja, S., Covell, M., Sukthankar, R.: Variable rate image compression with recurrent neural networks. In: ICLR (2016)
7. Toderici, G., Vincent, D., Johnston, N., Hwang, S.J., Minnen, D., Shor, J., Covell, M.: Full resolution image compression with recurrent neural networks. In: CVPR, pp. 5306–5314 (2017)
8. Johnston, N., Vincent, D., Minnen, D., Covell, M., Singh, S., Chinen, T., Hwang, S.J., Shor, J., Toderici, G.: Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks. arXiv preprint arXiv:1703.10114 (2017)
9. Ballé, J., Laparra, V., Simoncelli, E.P.: End-to-end optimized image compression. In: ICLR (2017)
10. Theis, L., Shi, W., Cunningham, A., Huszár, F.: Lossy image compression with compressive autoencoders. In: ICLR (2017)
11. Rippel, O., Bourdev, L.: Real-time adaptive image compression. In: ICML, pp. 2922–2930 (2017)
12. Jiang, F., Tao, W., Liu, S., Ren, J., Guo, X., Zhao, D.: An end-to-end compression framework based on convolutional neural networks. IEEE Trans. Circ. Syst. Video Technol. (2017). https://doi.org/10.1109/TCSVT.2017.2734838
13. Baig, M.H., Torresani, L.: Multiple hypothesis colorization and its application to image compression. Comput. Vis. Image Underst. (2017)
14. Prakash, A., Moran, N., Garber, S., DiLillo, A., Storer, J.: Semantic perceptual image compression using deep convolution networks. In: DCC, pp. 250–259 (2017)
15. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504–507 (2006)
16. Wong, C.W., Au, O.C., Lam, H.K.: Rate control using probability of non-zero quantized coefficients. In: ICME (2004)
17. Candes, E.J., Tao, T.: Decoding by linear programming. IEEE Trans. Inf. Theory **51**(12), 4203–4215 (2005)
18. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: ICML, pp. 807–814 (2010)
19. Schaefer, G., Stich, M.: UCID: an uncompressed color image database. In: Electronic Imaging 2004, International Society for Optics and Photonics, pp. 472–480 (2004)
20. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: convolutional architecture for fast feature embedding. In: ACM Multimedia, pp. 675–678. ACM (2014)
21. Said, A.: Introduction to arithmetic coding - theory and practice. Technical report HPL-2004-76, Hewlett Packard Laboratories Palo Alto (2004)