Signal Processing: Image Communication 80 (2020) 115678

Contents lists available at ScienceDirect

Signal Processing: Image Communication

journal homepage: www.elsevier.com/locate/image

Densely connected convolutional network block based autoencoder for panorama map compression ☆

Shengwei Wang [a], Hongkui Wang [a], Sen Xiang [b], Li Yu [a,*]

[a] School of Electron. Inform. & Commun., Huazhong Univ. of Sci. & Tech., Wuhan, 430074, China
[b] School of Inform. Sci. & Engn., Wuhan Univ. of Sci. & Tech., Wuhan, 430081, China

ARTICLE INFO

Keywords:
Autoencoder
Dense block
Neural network
Panorama map

ABSTRACT

As a novel virtual reality (VR) format, panorama maps are attracting increasing attention, while the compression of panorama images is still a concern. In this paper, a densely connected convolutional network block (dense block) based autoencoder is proposed to compress panorama maps. In the proposed autoencoder, dense blocks are specially designed to reuse feature maps and reduce redundancy of features. Meanwhile, a loss function, which imports a position-dependent weight item for each pixel, is proposed to train and adjust network parameters, in order to make the autoencoder fit to properties of panorama maps. Based on the proposed autoencoder and the weighted loss function, a greedy block-wise training scheme is also designed to avoid gradient vanishing problem and speed up training. During training process, the autoencoder is divided into several sub-nets. After each sub-net is trained separately, the whole network is fine-tuned to achieve the best performance. Experimental results demonstrate that the proposed autoencoder, compared with JPEG, saves up to 79.69 % bit rates, and obtains 7.27dB gain in BD-WS-PSNR or 0.0789 gain in BD-WS-SSIM. The proposed autoencoder also outperforms JPEG 2000, HEVC and VVC in both BD-WS-PSNR and BD-WS-SSIM. Meanwhile, subjective results show that the proposed autoencoder can recover details of panorama images, and reconstruct maps with high visual quality.

1. Introduction

As new advances in three dimensional (3D) technology, virtual reality (VR) is in great demand in many fields, such as education [1], manufacture [2], tourism [3] and design [4]. In order to provide complete immersive experiences to viewers, VR applications widely utilize panorama maps. Panorama images are also known as 360-degree images [5], which involve a 360-degree view of the scene captured from one single point. Meanwhile, panorama images can be projected to an internal surface of a sphere, and viewers can see any portion of the sphere with a head mounted display (HMD) device [6].

Compared with normal images, panorama maps require increased resolution (4K and beyond) to support 360-degree views of scenes. Meanwhile, panorama maps have special quality assessment methods, because images need to be projected to spheres instead of being watched directly. Thus, the growing popularity of VR applications brings new challenges in panorama map compression. Existing image coding standards, such as JPEG [7], JPEG 2000 [8], WebP [9] and BPG [10], are designed to compress traditional images, which cannot fully satisfy the needs of panorama maps. New codecs aiming at compressing panorama images are in great demand.

Recently, due to the development of deep learning, convolutional neural network (CNN) [11,12] has shown great potential in image compression. By the cascade of convolutional layers, CNNs are able to extract features and decompose images. During the decomposition, the dimension of images is reduced, and feature maps with small scale are generated. By quantizing and entropy coding small feature maps, images can be compressed efficiently.

Motivated by the observation above, in this paper, we propose a densely connected convolutional network block (dense block) based autoencoder to achieve end-to-end compression of panorama maps. Concretely, the proposed autoencoder consists of an encoding network and a decoding network. Both encoding and decoding networks are made up of several dense blocks, transition layers and convolutional layers. During the process of forward propagation, the features of panorama images are analyzed and extracted. Then, the extracted features are used to decompose and compress input images. In the encoding part, pooling layers are adopted to reduce the scale of input images, while sub-pixel layers [13], in the decoding part, are used to restore images to the original scale. To the best of our knowledge, it is the first time to design the autoencoder with dense blocks to compress panorama maps.

☆ No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.image.2019.115678.

* Corresponding author.
E-mail address: hustlyu@hust.edu.cn (L. Yu).

https://doi.org/10.1016/j.image.2019.115678
Received 14 November 2018; Received in revised form 23 October 2019; Accepted 24 October 2019
Available online 28 October 2019

Overall, the main contributions of our work can be summarized as follows:

- **A dense block based autoencoder.** A dense block based autoencoder is designed to achieve efficient end-to-end panorama image compression. In the proposed autoencoder, convolutional layers are used to analyze and extract features of images. The densely connection of each convolutional layer in dense blocks can decompose input panorama images more comprehensively, and reduce the redundancy of feature maps efficiently.
- **Cubic projection based block partition.** After researching a variety of spherical projection methods, the cubic projection is utilized to project the original panorama image to a cube. By further dividing and rotating, panorama images can be divided into 24 blocks with the same weight distribution. By the division, the autoencoder can compress images at block level, reducing the amount of memory usage and accelerating the compression speed.
- **A weighted loss function.** Panorama maps are projected to spheres to display, and pixels in different positions have different importance. Thus, a weighted loss function is proposed to optimize parameters of the autoencoder. In the proposed loss function, each pixel has a position-dependent weight item. With this weighed loss function, the autoencoder can adjust network parameters to be adapted to the importance of pixels.
- **Greedy block-wise training.** The proposed autoencoder cascades multiple dense blocks and convolutional layers. In order to avoid gradient vanishing problem and speed up convergence, a greedy block-wise training scheme is designed. Specifically, the whole autoencoder, during training, is divided into several sub-nets, and each sub-net is pre-trained solely. After that, the proposed autoencoder is initialized with the pre-trained parameters from sub-nets, and fine-tuned.

The remainder of this paper is organized as follows: In Section 2, related works are presented. Section 3 briefly introduces the autoencoder. The detail of our proposed dense block based autoencoder is shown in Section 4. Experimental results are presented in Section 5. Section 6 draws the conclusion.

2. Related works

Traditional image coding standards mainly rely on reducing redundancy to compress images. In order to reduce frequency domain redundancy, JPEG [7] and JPEG 2000 [14] adopt the discrete cosine transform (DCT) and the discrete wavelet transform (DWT) respectively to convert images from spatial domain to frequency domain. Besides frequency domain redundancy, spatial redundancy of maps can also be utilized. Correspondingly, webP [15] and BPG [10] adopt intra prediction methods to reduce spatial redundancy to further improve coding performance. Moreover, all-intra mode of HEVC/H.265 [16] and VVC/H.266 [17] can also be used to compress images. Although above standards have achieved high compression ratio, all of them are designed for normal images, not considering properties of panorama maps. A specialized codec for panorama maps is still in need.

Nowadays, CNNs have shown great promise in many fields. Beginning with image classification, Krizhevsky et al. [18] proposed AlexNet and won the ImageNet Competition. After that, Ren et al. [19] proposed a region proposal network (RPN), and merged the network with fast R-CNN into a single network for real-time object detection. In image denoising, Zhang et al. [20] proposed a residual network [21] based CNN to remove image noises, which can be used to solve Gaussian denoising, single image super-resolution, and JPEG image deblocking. Umehara et al. [22] proposed a super-resolution CNN (SRCNN) to directly realize an end-to-end mapping between low and high resolution images. A fine-tuned CNN is proposed by Radenovic et al. [23] to retrieval images from a large collection of unordered images. Besides, CNNs also achieve excellent performance in action recognition [24,25], salient detection [26], image forensics [27], and so on.

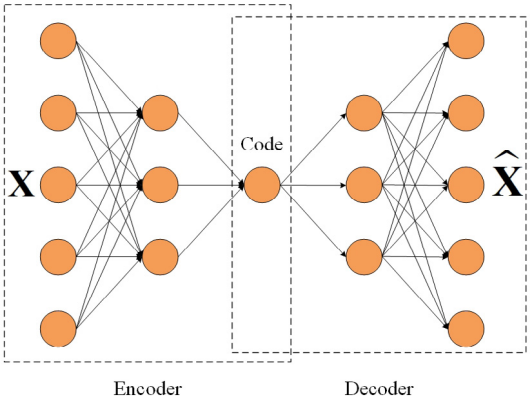


Fig. 1. The typical architecture of an autoencoder.

As for image compression, deep learning and CNNs also attract lots of attention. Toderici et al. [28,29] compared the property of different recurrent neuron units, and proposed an image compression method based on recurrent neural network (RNN) [30]. Extending Toderici's method, Johnston et al. [31] improved the RNN based autoencoder, and achieved better performance than JPEG, JPEG 2000 and webP. Moreover, Ballé et al. [32] theoretically analyzed the basic architecture of the CNN based image compression method, and proposed a joint model to optimize rate–distortion performance. Inspired by residual networks [21], Theis et al. [33] proposed an autoencoder architecture to realize lossy image compression. By designing a generative adversarial network (GAN), Agustsson et al. [34] achieved image compression in low bit-rate. Although above methods have implement efficient image compression, there is still much to be improved for coding panorama maps. Different from normal maps, weighted to spherically uniform PSNR (WS-PSNR) [35] is adopted to evaluate panorama image quality. Correspondingly, the autoencoder should fit to the assessment metric of panorama maps. Meanwhile, the autoencoder must be adapted to the large scale and the projection property of 360-degree image.

3. Autoencoder for compression

An autoencoder is an artificial neural network used for efficient coding [36,37]. A typical autoencoder is shown in Fig. 1. In the figure, the first three layers, including an input layer and two hidden layers, form an encoder. The encoder compresses the original signal $\mathbf{X}$ into a short code. The decoder, which consists of the last three layers, reconstructs $\mathbf{X}$ as $\hat{\mathbf{X}}$ from the code. Since the scale of the code is much smaller than $\mathbf{X}$, the autoencoder is able to compress $\mathbf{X}$.

The loss function of the autoencoder is usually defined as

$$L(\mathbf{X}, \hat{\mathbf{X}}) = ||\mathbf{X} - \hat{\mathbf{X}}||_2 \tag{1}$$

where L means the loss of the autoencoder. $\mathbf{X}$ and $\hat{\mathbf{X}}$ indicates the original and the reconstructed signal respectively. In implementation, the network parameters of the autoencoder are trained to minimize $L(\mathbf{X}, \hat{\mathbf{X}})$, in order to achieve the best performance.

4. Dense block based autoencoder for panorama map compression

In this section, the proposed method is explained in detail. The architecture of our proposed dense block based autoencoder is introduced in Section 4.1. The proposed weighted loss function aiming at compressing panorama maps is presented in Section 4.2. Section 4.3 explains the designed greedy block-wise training method for the proposed autoencoder.

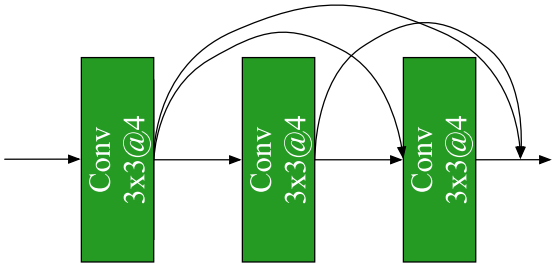


Fig. 2. The stereotype of a dense block. The notation $l \times l@c$ refers to $l \times l$ convolutional kernel with c channels.

4.1. Dense block based autoencoder

Considering that panorama images usually have large scale with abundant features, we specially designs the dense blocks to analyze and extract features of maps. By the densely connected convolutional layers in dense blocks, the proposed autoencoder can fully decompose input panorama images, and reuse extracted feature maps efficiently. Meanwhile, the redundancy of feature maps can be also reduced greatly. Moreover, the structure of dense connection requires less parameters to extract feature maps, and eases the gradient vanishing problem.

A stereotype of a dense block [38] is as Fig. 2 shows. The block comprises three convolutional layers, and each layer has four channels. Thus, we call it a 3-layer dense block with a growing rate of $k = 4$. In a dense block, the current convolutional layer connects the next layer. Meanwhile, the current layer also connects all of the remaining layers in the block.

In the proposed autoencoder, the dense block with a growing rate of $k = 32$ is designed as Fig. 3 shows. Each layer of the designed dense block contains three consecutive operations: batch normalization (BN) [39], a 3×3 convolution (Conv), and a rectified linear unit (ReLU) [40] activation function. In the encoder, the first dense block consists of 6 layers, while the second block has 12 layers. Since the input images of the second dense block, compared with the previous block, have been 2× down sampled, we double the number of layers of the second block to balance the complexity of each dense block. Similarly, the decoder has the same architecture. In designed dense blocks, any two layers are directly connected. Thus, feature maps from previous layers can be utilized multiple times in all of the latter layers. The reuse of feature maps makes the autoencoder reduce redundancy of features, and decompose images fully.

Besides the dense blocks, the other parts of the proposed autoencoder is described as follows.

Normalization: An input panorama image consists of R, G, B channels, in the range of [0, 255]. Since CNNs perform better for data ranging from 0 to 1, the proposed encoder normalizes each channel of input images as

$$I'(x, y) = \frac{I(x, y)}{255} \tag{2}$$

where $I(x, y)$ is the original pixel value at position (x, y), and $I'(x, y)$ is the normalized value. Correspondingly, the encoder and the decoder have normalization and denormalization layers respectively.

Convolutional layer: 3×3 convolutional kernels with stride 1 are used in convolutional layers to analyze images. Meanwhile, 1-padding is adopted to ensure that the scale of images does not change after convolution.

Transition layer: The transition layer is cascaded after the dense block, aiming at reducing the number of channels generated by the dense block. In the encoder, a transition layer is a convolutional layer, which has 3×3 convolutional kernels with 32 channels, while the transition layer has 128 channels in the decoder.

Maxpooling layer: Max pooling layers are utilized in the encoder to 2× down sample images. In max pooling layers, 2×2 filters with stride 2 are used, and the number of channels is also set as 32.

Subpixel layer: A subpixel layer can turn an input tensor with r^2c channels into a tensor with c channels, and extend the scale of tensor from l to rl [13], as Fig. 4 shows. In the decoder, subpixel layers are adopted to convert inputs with 128 channels to 32 channels, and achieve 2× up sampling.

Overall, in the proposed codec, input maps are in total max pooled four times, achieving 16× downsampling. Besides, quantization and entropy coding further compress images. Correspondingly, the decoder adopts entropy decoder, dequantization layer and four subpixel layers to recover images to the original scale. The reuse of feature maps in dense blocks reduces the redundancy of features, and decomposes input images more comprehensively, ensuring high compression ratio with excellent quality. Meanwhile, the dense blocks require less parameters to extract features, reducing the complexity of the proposed autoencoder.

4.2. Weighted loss function for the autoencoder

Original panorama maps are stored in equirectangular projection format, in which the spherical panorama image is projected to a rectangular map. Pixels in the equirectangular map have position-dependent weights, when the original map is recovered to a sphere for viewing. Different weights of pixels make it difficult to divide original maps to small blocks to code. Meanwhile, traditional loss functions, such as L_1 and L_2 loss, evaluate distortion of each pixel equally,

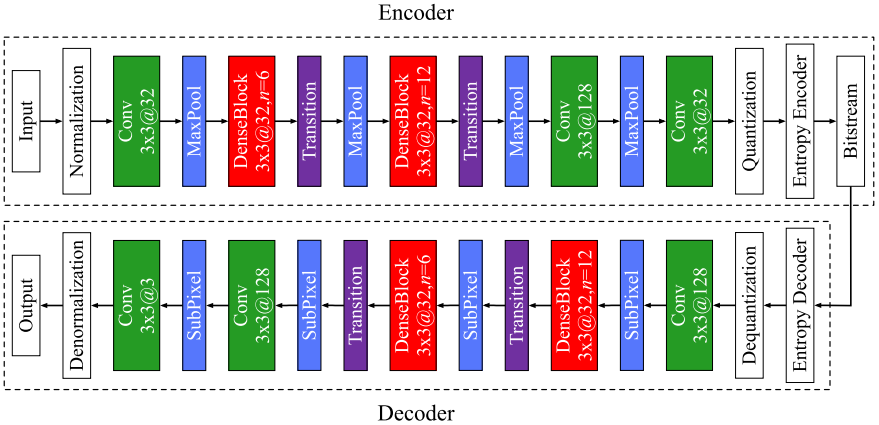


Fig. 3. The architecture of the proposed dense block based autoencoder. The notation $l \times l@c$ refers to $l \times l$ convolutional kernel with c channels. n indicates the number of layers in a dense block.

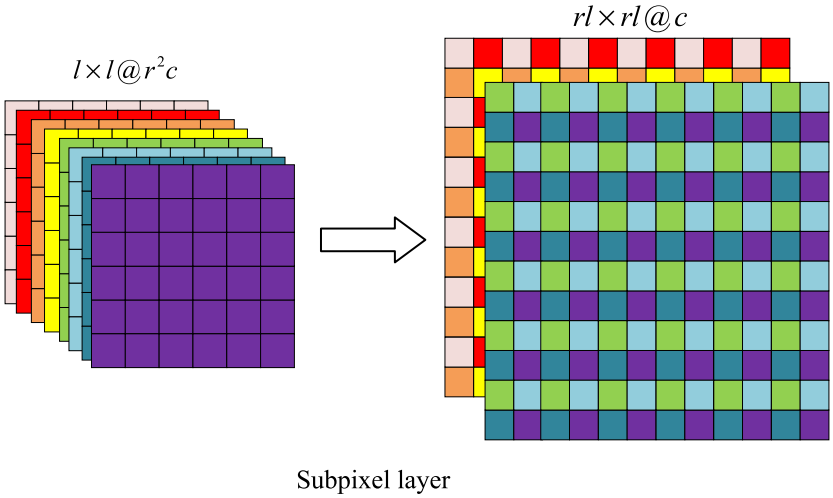


Fig. 4. The principle of subpixel layer. The notation $l \times l @ c$ refers to the scale of tensors is $l \times l$ with c channels. The figure shows that a subpixel layer converts a tensor with 8 channels $(r = 2, c = 2)$ into a tensor with 2 channels, and extends the scale of the tensor form l to $2l$.

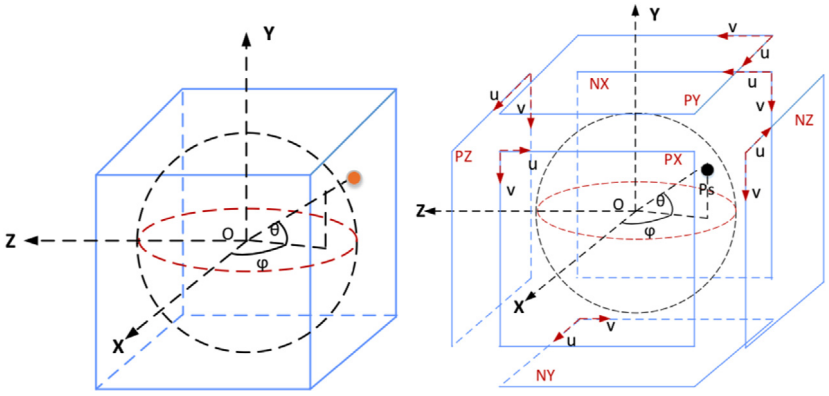


Fig. 5. The cubic projection. The original spherical panorama image is projected to six faces of a cube.

which is not reasonable to panorama maps. Thus, among a variety of projection methods, the proposed autoencoder utilizes cubic projection to transform the original panorama image to a cube. Based on cubic format, a novel weighted loss function is designed to train the proposed autoencoder.

In cubic projection, the spherical data is projected to a cube [41]. As Fig. 5 shows, the spherical data can be projected to a cube, and the details can be found in [42]. As an example, Fig. 6(a) is an equirectangular panorama map, Fig. 6(b) is the cubic image transformed from Fig. 6(a). Obviously, a cubic map can be easily divided into six sub-maps.

For each sub-map, pixels still have position-dependent weights. Assuming the resolution of a sub-map is $A \times A$, the weight can be calculated as

$$w_{cubic}(i,j) = (1 + \frac{d^2(i,j)}{r^2})^{-\frac{3}{2}} \tag{3}$$

where (i, j) is the position of the pixel, $r = A/2$ and w_{cubic} is the weight of the pixel in a cubic projection map. $d^2(i, j)$ means the distance between the face center and (i, j), and is defined as

$$d^2(i,j) = (i - \frac{A}{2})^2 + (j - \frac{A}{2})^2 \tag{4}$$

According to (3), the weight map of a panorama map with cubic format can be obtained as Fig. 7 shows. In the figure, brighter areas mean higher weights, while darker areas mean lower weights. It is obvious to see that six sub-maps have the same weight distribution. Then, each sub-map can be further divided into four blocks. Taking the top-left block as the reference, four blocks can also have the same weight distribution by rotating the remaining three blocks. By this way, a panorama image can be in total divided into 24 blocks with the same weight distribution. Then, the proposed autoencoder can encode images one block after another. By compressing images at block level, the proposed method can reduce the amount of memory usage and avoid out of memory, since panorama images usually have a very large scale, which makes it impossible to process the whole image directly. Meanwhile, the block level compression method can lower complexity greatly, accelerating the process speed.

According to the reference block, the weighted loss function is defined as

$$L(x, \hat{x}) = \sum_{i=1}^{r} \sum_{j=1}^{r} (1 + \frac{d^2(i,j)}{r^2})^{-\frac{3}{2}} (x(i,j) - \hat{x}(i,j))^2 \tag{5}$$

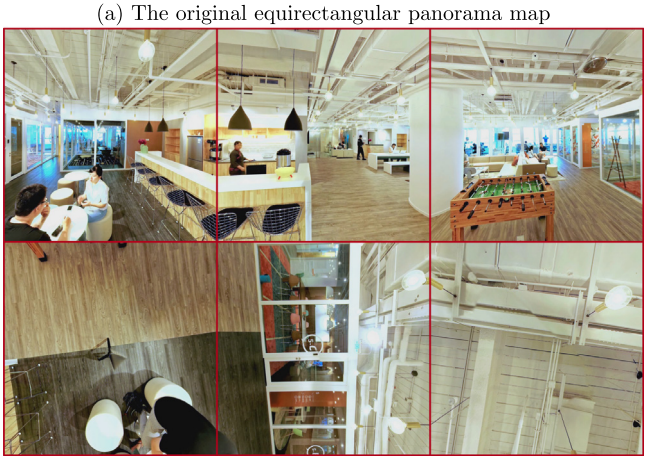
where x and $\hat{x}$ are the original input block and the output reconstructed block pixel respectively.

The proposed loss function makes the designed autoencoder reconstruct pixels of blocks with different quality according to the importance of each pixel. Then, the pixels are projected to the sphere, and ensure that the spherical map has the consistent quality, which provides natural immersive experiences to viewers with HMD devices.

4.3. Greedy block-wise training

In order to avoid gradient vanishing problem and accelerate convergence, a greedy block-wise training method is designed to train the

(a) The original equirectangular panorama map



(b) The projected cubic map

Fig. 6. The cubic projection of a panorama map. The equirectangular panorama image is projected to six faces of a cubic. (a) is the original equirectangular panorama map, (b) is the projected cubic map.

proposed autoencoder. The designed training method mainly includes two steps. In the first step, the proposed autoencoder is divided into several sub-nets, and each sub-net is solely trained. In the second step, the whole autoencoder is initialized by the trained parameters from sub-nets, and fine-tuned.

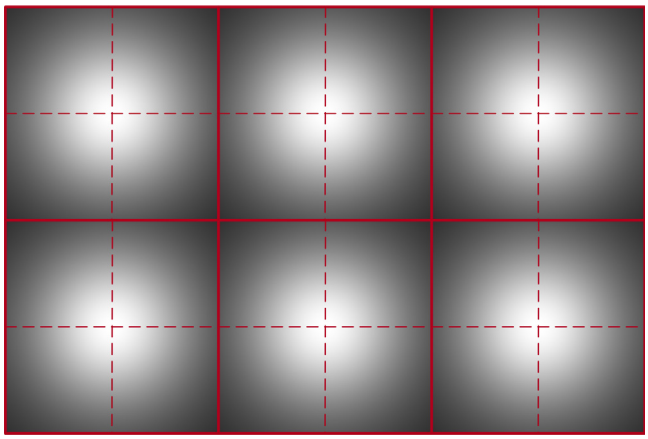


Fig. 7. The weight map of a panorama image with cubic format. Brighter areas mean higher weights, while darker areas mean lower weights.

In practice, the proposed autoencoder is divided into four sub-nets as Fig. 8 shows. In each sub-net, the layers in the encoder and the layers in the decoder are cascaded as one net. In the training process, each sub-net is treated as an independent network. When training one sub-net, the other sub-nets can be ignored and the parameters of the other sub-nets also cannot be changed. After four sub-nets are trained, all of the optimized parameters from sub-nets are used to initial the proposed autoencoder. Then, the whole autoencoder is fine-tuned to achieve better performance.

During the training process, different dataset is prepared for each sub-net. For sub-net 1, the dataset consists of the original panorama images, which are indicated as x_0. The dataset of sub-net 2 consists of x_1, which is the output of the encoding layers of the sub-net 1. After training sub-net 1, x_1 can be obtained. Similarly, x_2 and x_3 are the

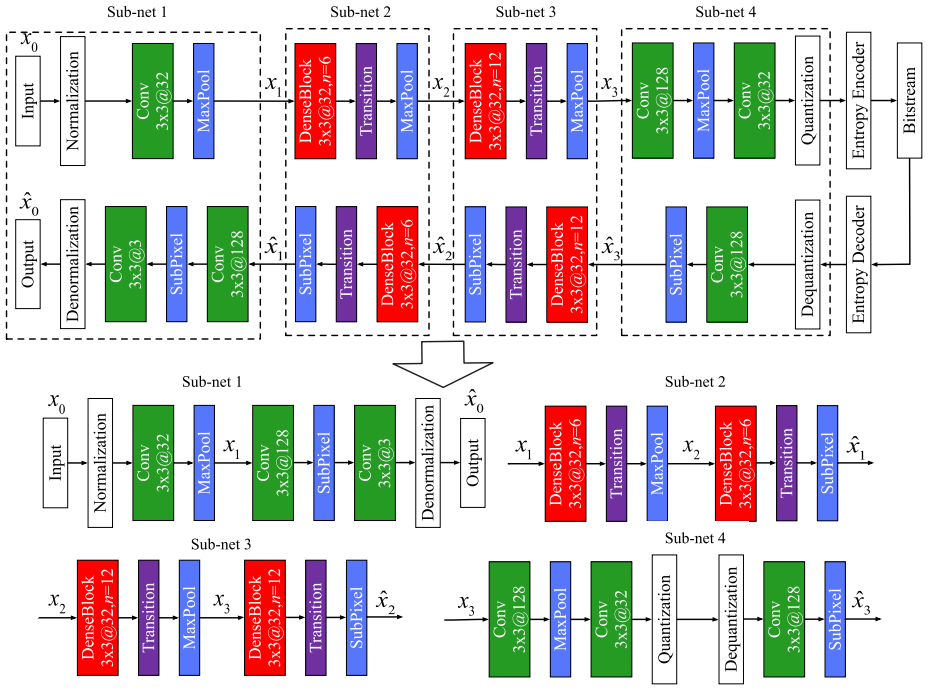
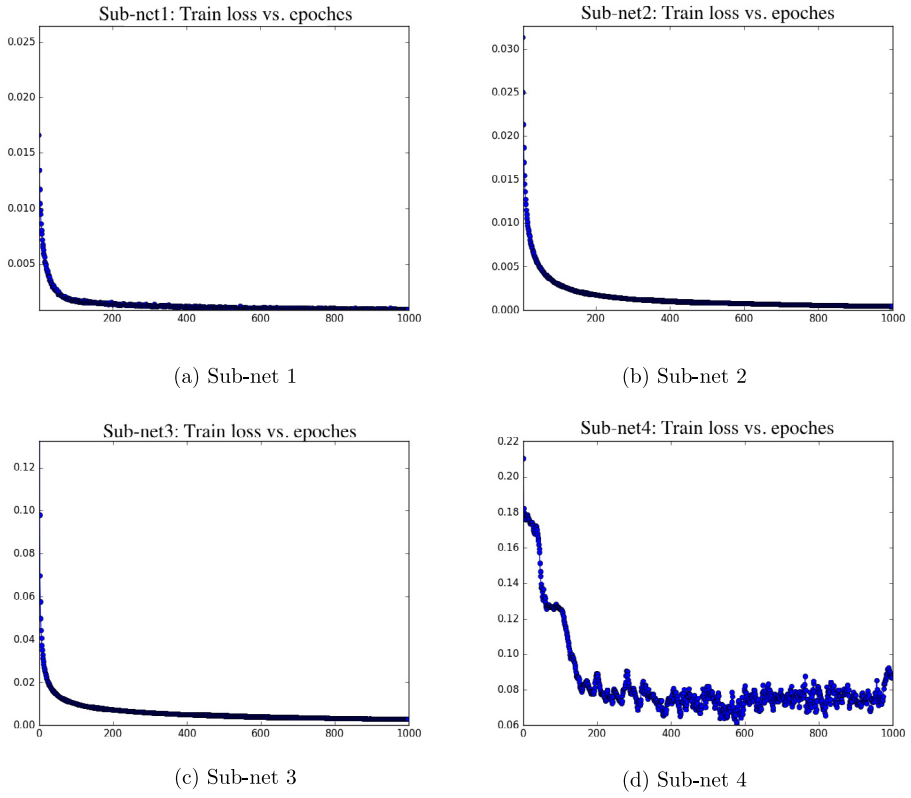


Fig. 8. The sub-net division of the proposed autoencoder for greedy block-wise training. The notation $l \times l@c$ refers to $l \times l$ convolutional kernel with c channels. n indicates the number of layers in a dense block.



(a) Sub-net 1 (b) Sub-net 2

(c) Sub-net 3 (d) Sub-net 4

Fig. 9. The loss curves of four sub-nets. x-axis is the number of epochs, y-axis is the training loss, and the loss function is the proposed weighted loss function.

datasets for sub-net 3 and sub-net 4 respectively. The loss function used in the training is the proposed weighted loss function. The purpose of training is to reduce the weighted distortion between x_i and $\hat{x}_1$ $(i = 0, 1, 2, 3)$ in each sub-net.

The proposed greedy block-wise training method decomposes the proposed autoencoder to four shallow sub-nets to train separately. This method efficiently avoids gradient vanishing problem in training a deep network. Meanwhile, training a shallow network is much easier and faster than training a deep network. Thus, the proposed method can reduce the complexity of training and accelerate convergence. Moreover, fine-tuning optimizes the parameters of the whole autoencoder based on pre-trained sub-nets, which improves the final coding performance.

Especially, the quantization layer of the proposed network is as follows

$$\hat{y} = \left[\frac{y}{q}\right] \tag{6}$$

where $\hat{y}$ is the quantized value of y, and q is the quantization step.

In back propagation, the gradient of the quantization function is 0 everywhere except at kq $(k = 0, \pm1, \pm2...)$. In order to train the network, the gradient of quantization layer is replaced in the back propagation process with the gradient of a smooth approximation r as

$$\frac{d\hat{y}}{dy} = \frac{d}{dy}\left[\frac{y}{q}\right] := \frac{d}{dy} r(\frac{y}{q}) \tag{7}$$

Empirically, we set r as

$$r(\frac{y}{q}) = \frac{y}{q} \tag{8}$$

This makes back propagation of quantization layer easy to implement, since the gradient of the quantization layer can be set as $\frac{1}{q}$ and passed. Similarly, the gradient of dequantization layer can be set as q.

5. Experimental results

In order to verify the performance of the proposed method, three experiments are conducted. The first part of the experiments shows the convergence curves of each sub-net and the whole proposed autoencoder, in order to demonstrate the effectiveness of the proposed greedy block-wise training method. The second part verifies the R-D performance of the proposed autoencoder. The third part compares the coding efficiency at different level, in order to show the performance of the proposed block partition scheme. The detailed results are presented in the following subsections.

5.1. Autoencoder training

In experiments, the training set consists of 16 panorama video sequences, such as 'Logo_Concert', 'AerialCity', 'Highway' and so on. For each sequence, 64 frames are randomly selected, and in total 1024 frames are adopted as training images. During training, the Adam [43] algorithm is used to backward propagate. The number of epoch is set as 1000, and the learning rate is 0.001.

The designed autoencoder is trained by the proposed greedy block-wise training method. Firstly, four sub-nets are trained. The loss curves of four sub-nets is shown in Fig. 9. From the figure, it is obvious to see that the sub-nets converge fast in the training due to the shallow structures. Especially, only sub-net 4 is shaking near the convergence point, since there is quantization layer in sub-net 4, which increases distortion.

Then, the proposed autoencoder is initialed by the pre-trained four sub-nets, and fine-tuned. The loss curve is also show in Fig. 10. In the figure, the blue line with dots is the loss curve of fine-tuning. It is obvious to see that the loss of the autoencoder starts from a small value, and converges after several epochs quickly. Meanwhile, the cascade of dense blocks decreases the loss shaking brought by quantization, compared with the loss curve of the sub-net 4. Compared with the proposed training method, the green line with triangles, which indicates the loss curve of the traditional training method, is still shaking around a high value. Due to the gradient vanishing problem, the original deep network is hard to converge. Thus, the experimental results demonstrate

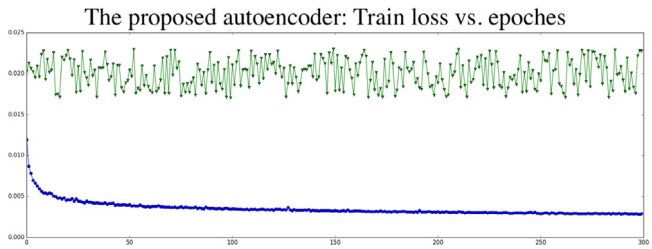


Fig. 10. The comparison of loss curves of the whole proposed autoencoder between the proposed training method and the traditional training method. x-axis is the number of epochs, y-axis is the training loss, and the loss function is the proposed weighted loss function. The blue line with dots indicates the proposed method, and the green line with triangles indicates the traditional training method.

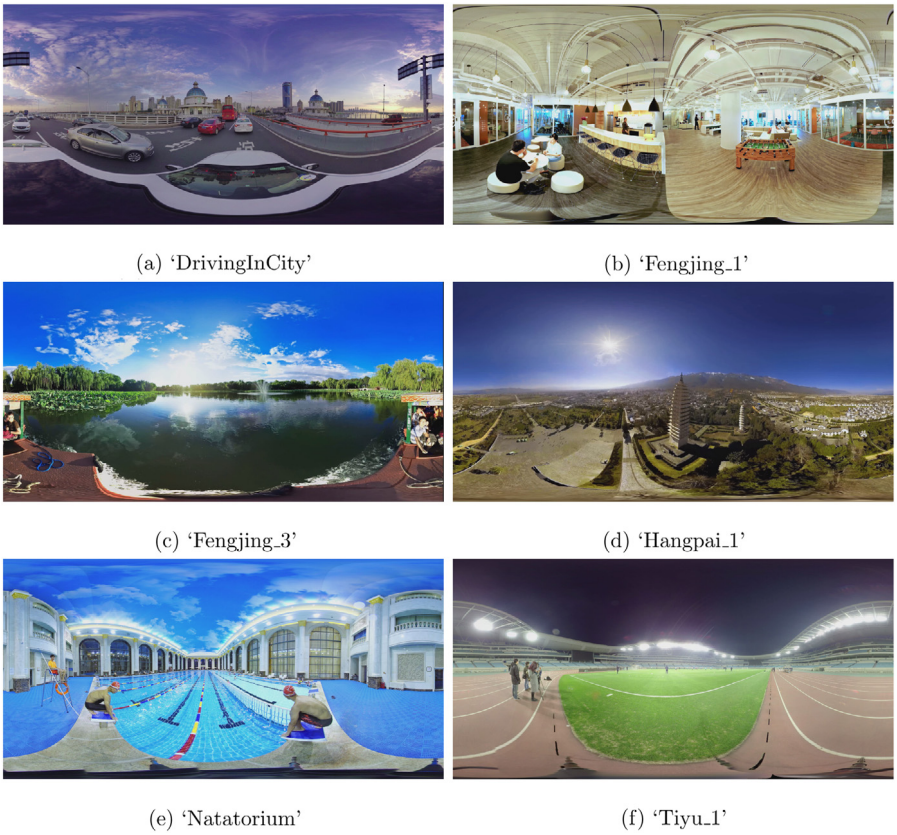


Fig. 11. The six test images used in the experiments to evaluate the R-D performance of each method.

that the proposed greedy block-wise training method can make the network converge fast, accelerating training speed greatly. Meanwhile, the proposed training method divides a deep network to several shallow networks to train, avoiding the gradient vanishing problem.

5.2. R-D performance

Six panorama maps are chosen as the test images in the experiments, in order to evaluate the rate–distortion (R-D) performance of the proposed autoencoder. The detailed images are shown in Fig. 11. All of them are the original panorama images with equirectangular format, and the resolution of them is 4096 × 2048. In the experiments, the test images are stored as lossless BMPs in advance, and the images are uncompressed.

In the experiments, the proposed autoencoder is compared with JPEG [7], JPEG 2000 [14], HEVC/H.265 [16], VVC/H.266 [17]. Among these methods, JPEG and JPEG 2000 are traditional image coding standards, which are mainly based on DCT and DWT respectively. HEVC and VVC are novel video coding standards. Besides transforming, HEVC and VVC also adopt intra prediction to reduce spatial redundancy of images. In the experiments, HM 16.16 is selected as HEVC reference software, and VVC reference software is VTM 0.2.

Meanwhile, the proposed method is also compared with recently published neural network based methods, including Ballé's method [32], Toderici's method [29], and Agustsson's method [34]. Among these methods, Ballé [32] designed a standard CNN based autoencoder to compress images. In his method, traditional convolutional layers are used to extract feature maps without any special design or connection. Instead of CNN, Toderici [29] imported RNN, and designed an RNN based autoencoder. In his method, one CNN block and three RNN blocks are cascaded in the encoder, and the decoder has the similar architecture. In Agustsson's generative adversarial network (GAN) based method [34], besides traditional image compression, a generative network is specially designed to enhance the local patches of images, improving the compression quality greatly. Different from above methods, the proposed method adopts dense connections, and specially designs dense blocks for better compression of panorama images. Moreover, considering that all of the above neural network

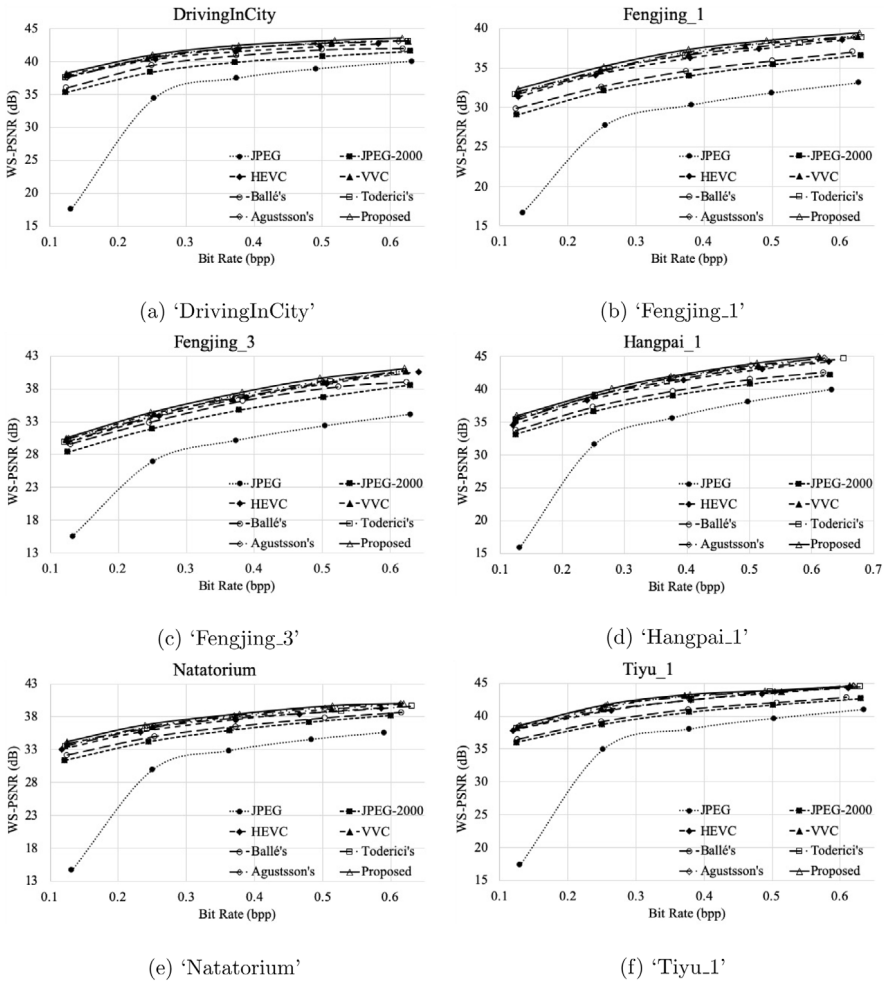


Fig. 12. Comparison of eight methods with respect to WS-PSNR. x-axis indicates bit rate to encode panorama maps, y-axis indicates WS-PSNR of compressed maps.

based methods are not integrated with entropy coders, the context-adaptive binary arithmetic coder (CABAC) is used as entropy coding for a fair comparison in experiments. Meanwhile, the proposed weighted loss function is also applied in the training of above methods for fairness.

In experiments, five bit-rate points, including 0.125 bits per pixel (bpp), 0.250 bpp, 0.375 bpp, 0.500 bpp, and 0.625 bpp, are selected as benchmark. Original equirectangular panorama images are used to evaluate coding quality. The quality metric for panorama maps is the average WS-PSNR of R, G, B channels of the image. WS-PSNR of each channel is calculated as

$$\mathrm{WS-PSNR} = 10\log\left(\frac{255^2}{\mathrm{WMSE}}\right) \tag{9}$$

In (9), WMSE is defined as

$$\mathrm{WMSE} = W(i,j)\sum_{i=0}^{height-1}\sum_{j=0}^{width-1}(x(i,j)-\hat{x}(i,j))^2 w_{rect}(i,j) \tag{10}$$

where *width* and *height* indicate the size of the map. $x(i,j)$ and $\hat{x}(i,j)$ indicate the original pixel value and the reconstructed pixel value at position (i,j) respectively. $w_{rect}(i,j)$ is the weight of the pixel at position (i,j). $W(i,j)$ is defined as

$$W(i,j) = \frac{1}{\sum_{i=0}^{height-1}\sum_{j=0}^{width-1} w_{rect}(i,j)} \tag{11}$$

and $w_{rect}(i,j)$ is defined as

$$w_{rect}(i,j) = \cos\left(\left(i - \frac{height}{2} + \frac{1}{2}\right)\cdot\frac{\pi}{height}\right) \tag{12}$$

Similarly, we also adopt WS-SSIM as the quality assessment metric to evaluate performance of each method. For each SSIM block, the weight is depicted as the weight of the center pixel of the block, and is calculated by (12).

In experiments, all of the six test maps are encoded by each method at or near five given bit-rate points. R-D curves of eight methods are shown in Figs. 12 and 13. In the figures, 'Ballé's' indicates CNN-based codec proposed by Ballé et al. [32], 'Toderici's' means the RNN-based codec proposed by Toderici et al. [29], 'Agustsson's' shows the GAN-based codec proposed by Agustsson et al. [34], and 'Proposed' presents our proposed dense block based autoencoder. Fig. 12 shows the WS-PSNR performance, while Fig. 13 shows the WS-SSIM performance. From the figures, it is obvious to see that our method outperforms other methods with the highest WS-PSNR and WS-SSIM at the same bit-rate.

Moreover, we also calculate the Bjontegaard-Delta (BD)-WS-PSNR, BD-WS-SSIM and BD-rate by taking JPEG as the baseline. The detailed results are shown in Table 1. In the table, the first and second rows of each map are calculated according to the rate-WS-PSNR curves in Fig. 12, while the third and fourth rows are calculated by the rate-WS-SSIM curves in Fig. 13. A positive value for BD-WS-PSNR and BD-WS-SSIM indicates quality improvement of the compared codec than JPEG at the same bit-rate. A negative value for BD-rate indicates bit-rate saving of the compared codec than JPEG at the same quality. Results in Table 1 are consistent with Figs. 12 and 13, and the proposed autoencoder achieves the best R-D performance among eight methods. Especially, compared with JPEG, the proposed method reaches up to 79.69% bit saving, 7.27 dB gain in BD-WS-PSNR, and 0.0789 gain in BD-WS-SSIM.

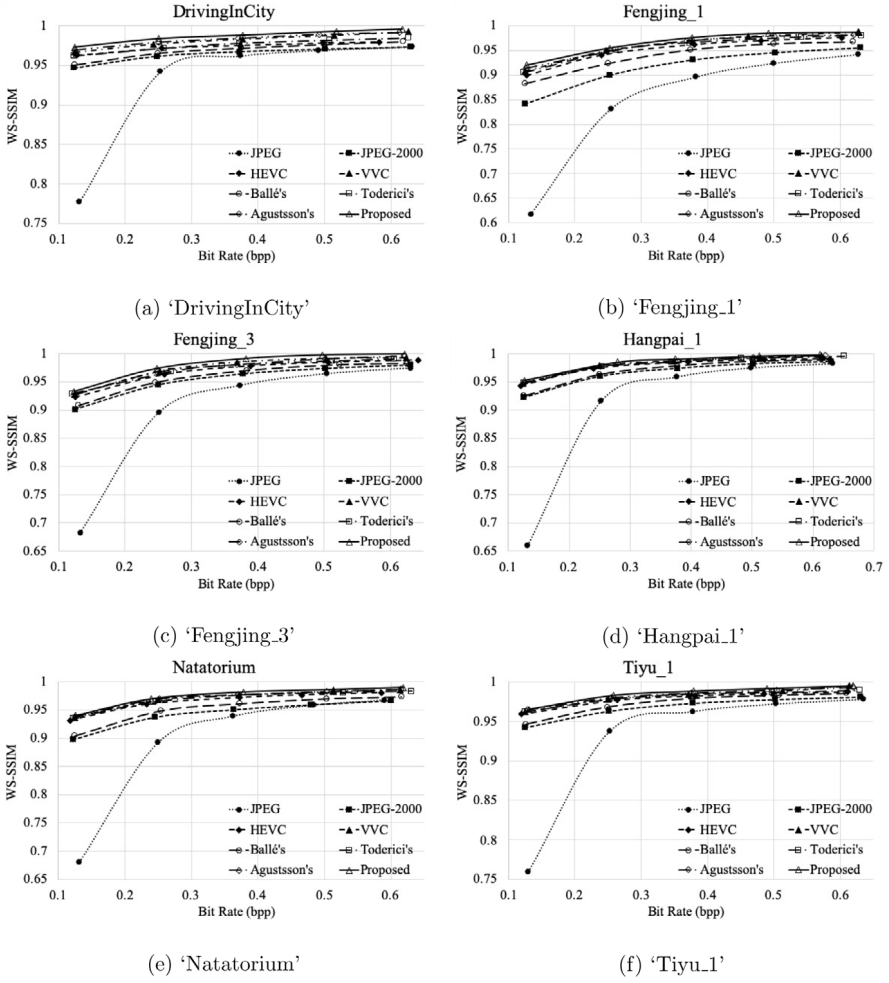


Fig. 13. Comparison of eight methods with respect to WS-SSIM. *x*-axis indicates bit rate to encode panorama maps, *y*-axis indicates WS-SSIM of compressed maps.

Table 1
Coding efficiency comparison of eight methods. In each image, the first and second rows are calculated according to the rate-WS-PSNR curves, while the third and fourth rows are calculated according to the rate-WS-SSIM curves.

Image name	Measure	Image codecs							
		JPEG	JPEG-2000	HEVC	VVC	Ballé's	Toderici's	Agustsson's	Proposed
DrivingInCity	BD-rate (%)	–	−42.33	−64.67	−70.52	−47.21	−65.27	−68.14	**−72.17**
	BD-WS-PSNR (dB)	–	2.47	4.22	4.46	3.38	4.51	4.57	**4.92**
	BD-rate (%)	–	−18.77	−53.47	−69.00	−36.00	−56.14	−81.62	**−87.62**
	BD-WS-SSIM	–	0.0062	0.0152	0.0227	0.0109	0.0172	0.0239	**0.0276**
Fengjing_1	BD-rate (%)	–	−50.29	−68.94	−73.78	−57.03	−68.35	−70.14	**−79.69**
	BD-WS-PSNR (dB)	–	3.75	6.17	6.46	4.38	6.38	6.62	**6.96**
	BD-rate (%)	–	−32.30	−61.87	−66.41	−41.65	−65.54	−77.23	**−91.93**
	BD-WS-SSIM	–	0.0357	0.0672	0.0751	0.0559	0.0672	0.0741	**0.0789**
Fengjing_3	BD-rate (%)	–	−46.81	−58.21	−60.75	−52.45	−59.01	−59.30	**−62.23**
	BD-WS-PSNR (dB)	–	4.60	6.43	6.65	5.71	6.71	6.87	**7.27**
	BD-rate (%)	–	−27.62	−46.52	−53.02	−25.89	−52.31	−52.38	**−59.12**
	BD-WS-SSIM	–	0.0211	0.0337	0.0409	0.0260	0.0367	0.0392	**0.0469**
Hangpai_1	BD-rate (%)	–	−35.13	−52.67	−55.85	−40.84	−54.80	−54.75	**−56.91**
	BD-WS-PSNR (dB)	–	3.37	5.57	6.09	4.04	5.79	6.10	**6.24**
	BD-rate (%)	–	−23.27	−46.31	−53.24	−25.80	−49.27	−52.40	**−60.80**
	BD-WS-SSIM	–	0.0160	0.0267	0.0301	0.0205	0.0287	0.0300	**0.0302**
Natatorium	BD-rate (%)	–	−43.96	−62.42	−63.77	−50.09	−63.84	−67.17	**−76.82**
	BD-WS-PSNR (dB)	–	3.08	4.62	5.01	3.53	4.99	5.20	**5.37**
	BD-rate (%)	–	−13.22	−51.87	−58.31	−28.76	−57.53	−60.99	**−63.97**
	BD-WS-SSIM	–	0.0132	0.0341	0.0379	0.0218	0.0369	0.0383	**0.0423**
Tiyu_1	BD-rate (%)	–	−38.59	−58.05	−58.32	−44.84	−60.46	−61.79	**−67.29**
	BD-WS-PSNR (dB)	–	2.55	4.51	4.98	2.98	4.63	4.88	**5.12**
	BD-rate (%)	–	−29.29	−58.29	−63.47	−47.01	−61.98	−63.63	**−67.54**
	BD-WS-SSIM	–	0.0112	0.0201	0.0234	0.0172	0.0231	0.0242	**0.0266**

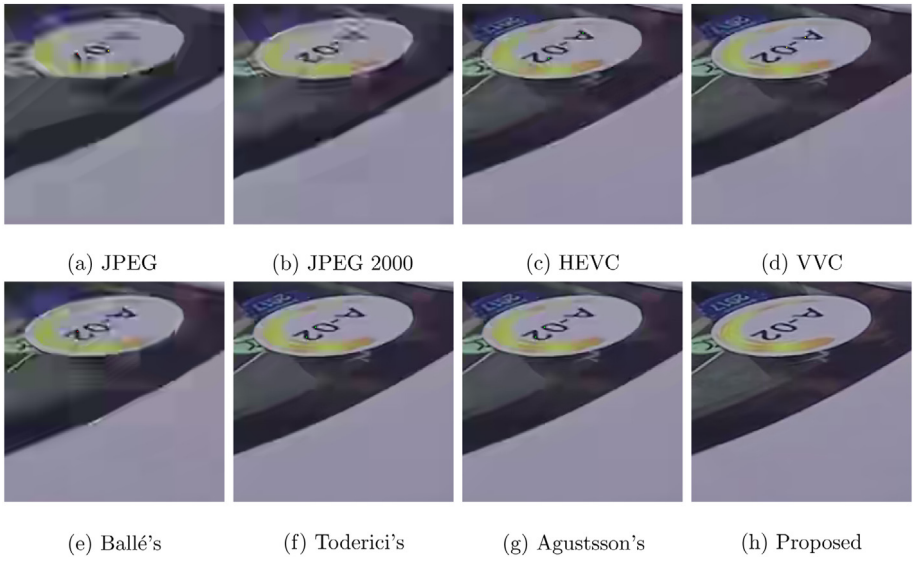


Fig. 14. Image patches of 'DrivingInCity' produced by eight methods at 0.375 bpp. The letters on the label is more clear in the proposed method.

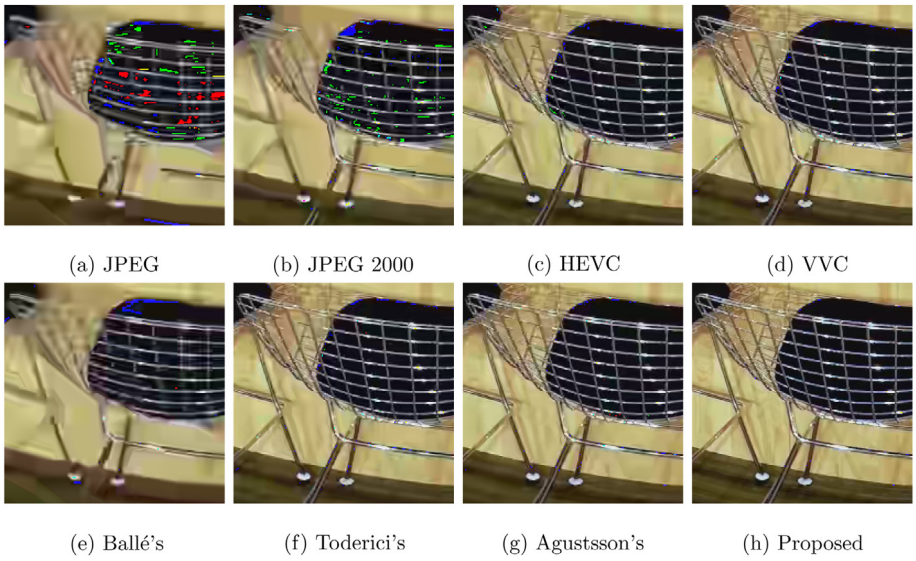


Fig. 15. Image patches of 'Fengjing_1' produced by eight methods at 0.625 bpp. The proposed method has the least noisy points.

In addition, subjective quality results of the six test images are also shown in Figs. 14, 15, 16, 17, 18, and 19. In the figures, the patches produced by JPEG still have blocking artifacts. The patches produced by JPEG-2000 and Ballé's method [32] also have obvious blurring. Among eight methods, the proposed autoencoder can greatly reduce the noisy points, and retain the detailed textures of images, providing better experiences to viewers. Specially, the proposed method can recover the letters on the label precisely as Fig. 14 shows. In Figs. 15 and 16, the patches produced by the proposed method have the least noisy points, and the texture of the patches are retained well. The outline of the tower in Fig. 17 and the texture on the wall in Fig. 18 are clearer to see in the patches produced by the proposed method. Meanwhile, the overcoat has more distinct patterns in the patch produced by the proposed method as Fig. 19 shows.

Compared with other methods, the proposed autoencoder specially designs dense blocks for panorama maps. By fully extracting and reusing feature maps, the dense block can decompose panorama images more comprehensively, and obtain abundant detail features, improving the coding quality of the proposed autoencoder. Further, the cascade of multiple dense blocks ensure that the input panorama images can be fully analyzed and decomposed to achieve a high compression ratio. Moreover, the proposed weighted loss function guides training, and adjusts network parameters to fit to the properties of panorama maps for better coding quality. Thus, the proposed autoencoder, compared with other methods, can compress and reconstruct panorama maps in a low bit rate with high quality.

5.3. Compression at different levels

Finally, in order to evaluate the performance of the proposed block partition scheme, the proposed autoencoder is used to compress panorama images at two different levels in the experiments. One level is to compress the whole panorama images directly, the other one is to compress blocks of images which are divided by the proposed cubic projection based partition scheme. Specially, when training the

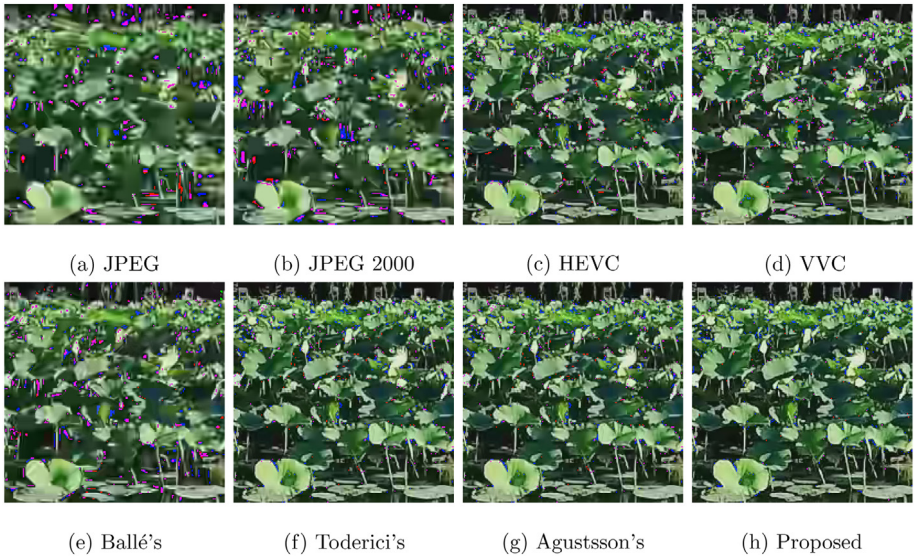


Fig. 16. Image patches of 'Fengjing_3' produced by eight methods at 0.500 bpp. The proposed method has the least noisy points, and the lotus leaves are clearer.

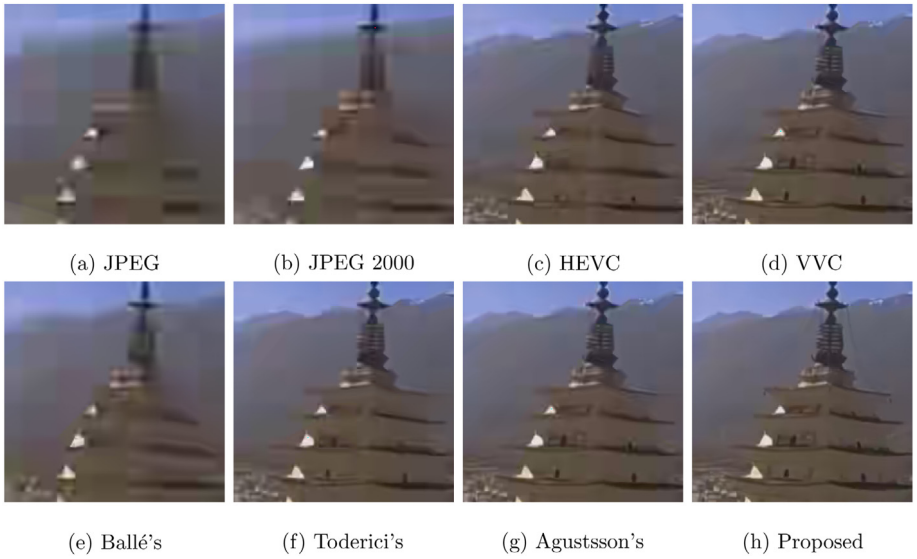


Fig. 17. Image patches of 'Hangpai_1' produced by eight methods at 0.250 bpp. The outline of the tower is more distinct in the proposed method.

autoencoder at image-level, the proposed weighted loss function is also extended to fit the whole image for a fair comparison.

In the experiments, six test panorama images are used to evaluate the performance of the proposed block partition scheme. Taking 0.250 bpp bit-rate as an example, the detailed results is shown in Table 2. In the table, 'Bpp' shows the bit-rate of coding image. 'Memory' and 'Time' respectively indicate the space consumption and coding time. 'WS-PSNR' and 'WS-SSIM' shows the quality of coded panorama images. For each image in the table, the first row is the results generated by image-level compression, and the second row is the results generated by block-level compression.

From the table, it is obvious to see that coding panorama images at two different level can achieve almost the same quality. In addition, compressing image at block-level, compared with image-level, only consumes half the time. Meanwhile, coding at block-level can save much memory space. Thus, the experimental results demonstrate that the proposed cubic projection based block partition scheme can save memory and lower coding complexity efficiently.

6. Conclusion

In this paper, a dense block based autoencoder aiming at panorama map compression is proposed. In the proposed autoencoder, we designed dense blocks to extract and reuse features of images, in order to reduce redundancy of features and improve coding efficiency. Moreover, a weighted loss function was constructed to train the autoencoder, according to the property of panorama images. A greedy block-wise training scheme was also proposed, in order to avoid gradient vanishing problem and accelerate training. The experimental results demonstrated that the proposed autoencoder, compared with JPEG, can save up to 79.69% bit rates, and obtain 7.27 dB gain in BD-WS-PSNR or 0.0789 gain in BD-WSSSIM. Compared with JPEG 2000, HEVC and VVC, the proposed method also achieved better performance. In addition, subject results proved that the proposed autoencoder can recover details of panorama maps, and reconstruct images with high visual quality.

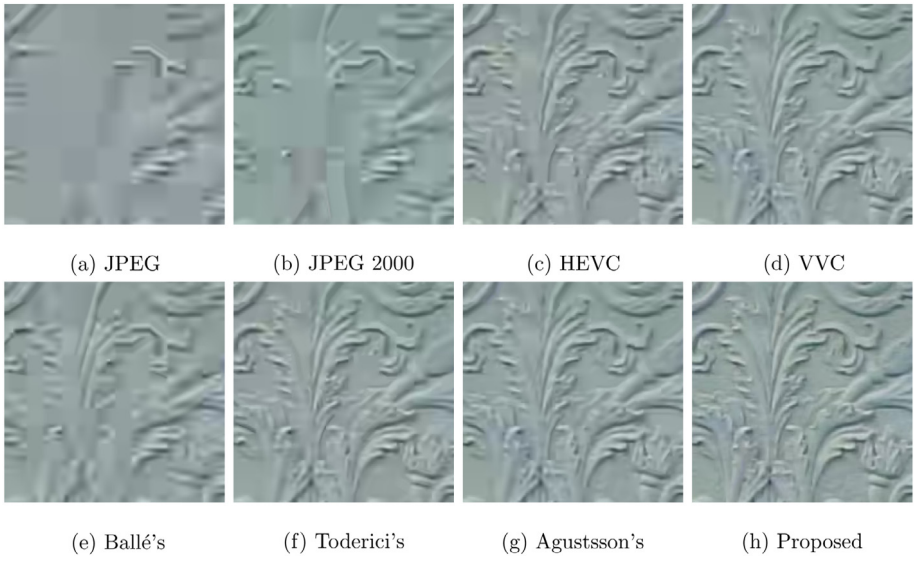


Fig. 18. Image patches of 'Natatorium' produced by eight methods at 0.375 bpp. The patterns on the wall are more distinct in the proposed method.

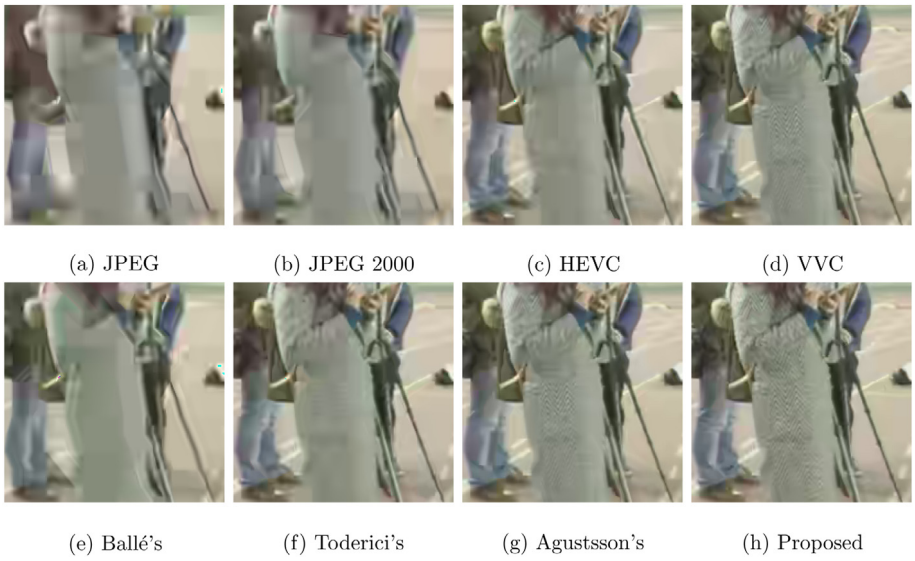


Fig. 19. Image patches of 'Tiyu_1' produced by eight methods at 0.250 bpp. The patterns on the overcoat are more distinct in the proposed method.

Table 2
Compression performance comparison at two different levels. In each image, the first row is the results generated by image-level compression. The second row is the results generated by block-level compression.

Image name	Level	Bpp	Memory (MB)	Time (s)	WS-PSNR (dB)	WS-SSIM
DrivingInCity	image	0.253	10843	4.78	41.02	0.9840
	block	0.251	1821	2.31	41.01	0.9836
Fengjing_1	image	0.252	10843	4.83	35.19	0.9551
	block	0.253	1821	2.47	35.18	0.9548
Fengjing_3	image	0.250	10843	4.69	34.36	0.9729
	block	0.248	1821	2.26	34.38	0.9734
Hangpai_1	image	0.273	10843	4.95	40.10	0.9849
	block	0.279	1821	2.39	40.11	0.9849
Natatorium	image	0.241	10843	4.87	36.72	0.9699
	block	0.239	1821	2.62	36.72	0.9698
Tiyu_1	image	0.256	10843	4.81	41.78	0.9831
	block	0.258	1821	2.57	41.79	0.9832

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 61871437, in part by the Natural Science Foundation of Hubei Province under Grant 2019CFA022, 2017CFB348, in part by the Research Foundation of Hubei Educational Committee under Grant Q20171106, and in part by the Research Foundation for Young Scholars of WUST under Grant 2017xz008.

References

[1] L. Freina, M. Ott, A literature review on immersive virtual reality in education: state of the art and perspectives, in: Int. Sci. Conf. E-Learning & Software for Educ., Vol. 1, "Carol I" National Defence University, 2015, p. 133.

[2] A. Renner, J. Holub, S. Sridhar, G. Evans, E. Winer, A virtual reality application for additive manufacturing process training, in: ASME 2015 Int. Design Eng. Tech. Conf. & Comput. & Inform. in Eng. Conf., American Soc. Mech. Engineers, 2015, http://dx.doi.org/10.1115/DETC2015-47807, V01AT02A033–V01AT02A033.

[3] Y.C. Huang, K.F. Backman, S.J. Backman, L.L. Chang, Exploring the implications of virtual reality technology in tourism marketing: An integrated research framework, Int. J. Tour. Res. 18 (2) (2016) 116–128, http://dx.doi.org/10.1002/JTR.2038.

[4] J.Q. Coburn, I. Freeman, J.L. Salmon, A review of the capabilities of current low-cost virtual reality technology and its potential to enhance the design process, J. Comput. Inf. Sci. Eng. 17 (3) (2017) 031013, http://dx.doi.org/10.1115/1.4036921.

[5] J. Ling, K. Zhang, Y. Zhang, D. Yang, Z. Chen, A saliency prediction model on 360 degree images using color dictionary based sparse representation, Signal Process., Image Commun. http://dx.doi.org/10.1016/J.IMAGE.2018.03.007.

[6] Z. Chen, Y. Li, Y. Zhang, Recent advances in omnidirectional video coding for virtual reality: Projection and evaluation, Signal Process. 146 (2018) 66–78, http://dx.doi.org/10.1016/J.SIGPRO.2018.01.004.

[7] G.K. Wallace, The JPEG still picture compression standard, IEEE Trans. Consum. Electron. 38 (1) (1992) xviii–xxxiv, http://dx.doi.org/10.1109/30.125072.

[8] A. Skodras, C. Christopoulos, T. Ebrahimi, The JPEG 2000 still image compression standard, IEEE Signal Process. Mag. 18 (5) (2000) 36–58, http://dx.doi.org/10.1109/79.952804.

[9] G. Ginesu, M. Pintus, D.D. Giusto, Objective assessment of the WebP image coding algorithm, Signal Process., Image Commun. 27 (8) (2012) 867–874, http://dx.doi.org/10.1016/J.IMAGE.2012.01.011.

[10] U. Albalawi, S.P. Mohanty, E. Kougianos, Energy-efficient design of the secure better portable graphics compression architecture for trusted image communication in the IoT, in: IEEE Comput. Soc. Annu. Symp. VLSI, IEEE, 2016, pp. 302–307, http://dx.doi.org/10.1109/ISVLSI.2016.21.

[11] H.H. Aghdam, E.J. Heravi, Convolutional neural networks, in: Guide to Convolutional Neural Networks, Springer, 2017, pp. 85–130, http://dx.doi.org/10.1007/978-3-319-57550-6.

[12] N. Ketkar, Convolutional neural networks, in: Deep Learning with Python, Springer, 2017, pp. 63–78, http://dx.doi.org/10.1007/978-1-4842-2766-4.

[13] W. Shi, J. Caballero, F. Huszár, J. Totz, A.P. Aitken, R. Bishop, D. Rueckert, Z. Wang, Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network, in: Proc. IEEE Conf. Comput. Vision & Pattern Recognition, 2016, pp. 1874–1883, http://dx.doi.org/10.1109/CVPR.2016.207.

[14] M.W. Marcellin, M.J. Gormish, A. Bilgin, M.P. Boliek, An overview of JPEG-2000, in: Proc. Data Compression Conf., IEEE, 2000, pp. 523–541, http://dx.doi.org/10.1109/DCC.2000.838192.

[15] V. Agababov, M. Buettner, V. Chudnovsky, M. Cogan, B. Greenstein, S. McDaniel, M. Piatek, C. Scott, M. Welsh, B. Yin, Flywheel: Google's data compression proxy for the mobile web., in: NSDI, vol. 15, 2015, pp. 367–380.

[16] G.J. Sullivan, J. Ohm, W.-J. Han, T. Wiegand, Overview of the high efficiency video coding (HEVC) standard, IEEE Trans. Circuits Syst. Video Technol. 22 (12) (2012) 1649–1668, http://dx.doi.org/10.1109/TCSVT.2012.2221191.

[17] Versatile Video Coding (Draft 2), document JVET-K1001-v7, in: document JVET-K1001-v7, Joint Video Experts Team (JVET) of ITU-T SG 16 WP3 and ISO/IEC JTC 1/SC 29/WG 11, 11th Meeting, Ljubljana, 2018.

[18] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Inform. Process. Syst, 2012, pp. 1097–1105, http://dx.doi.org/10.1145/3065386.

[19] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: Towards real-time object detection with region proposal networks, IEEE Trans. Pattern Anal. Mach. Intell. 39 (6) (2017) 1137–1149, http://dx.doi.org/10.1109/TPAMI.2016.2577031.

[20] K. Zhang, W. Zuo, Y. Chen, D. Meng, L. Zhang, Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising, IEEE Trans. Image Process. 26 (7) (2017) 3142–3155, http://dx.doi.org/10.1109/TIP.2017.2662206.

[21] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proc. IEEE Conf. Comput. Vision & Pattern Recognition, 2016, pp. 770–778, http://dx.doi.org/10.1109/CVPR.2016.90.

[22] K. Umehara, J. Ota, N. Ishimaru, S. Ohno, K. Okamoto, T. Suzuki, N. Shirai, T. Ishida, Super-resolution convolutional neural network for the improvement of the image quality of magnified images in chest radiographs, in: Medical Imaging 2017: Image Process., in: Int. Soci. for Optics & Photonics, vol. 10133, 2017, p. 101331P, http://dx.doi.org/10.1117/12.2249969.

[23] F. Radenović, G. Tolias, O. Chum, CNN image retrieval learns from BoW: Unsupervised fine-tuning with hard examples, in: European Conf. Comput. Vision, Springer, 2016, pp. 3–20, http://dx.doi.org/10.1007/978-3-319-46448-0_1.

[24] G. Varol, I. Laptev, C. Schmid, Long-term temporal convolutions for action recognition, IEEE Trans. Pattern Anal. Mach. Intell. http://dx.doi.org/10.1109/TPAMI.2017.2712608.

[25] S. Yan, J.S. Smith, W. Lu, B. Zhang, Hierarchical multi-scale attention networks for action recognition, Signal Process., Image Commun. 61 (2018) 73–84, http://dx.doi.org/10.1016/J.IMAGE.2017.11.005.

[26] M.T. Islam, S.M. Rahman, M.O. Ahmad, M. Swamy, Mixed gaussian-impulse noise reduction from images using convolutional neural network, Signal Process., Image Commun. 68 (2018) 26–41, http://dx.doi.org/10.1016/J.IMAGE.2018.06.016.

[27] J.-S. Park, H.-G. Kim, D.-G. Kim, I.-J. Yu, H.-K. Lee, Paired mini-batch training: A new deep network training for image forensics and steganalysis, Signal Process., Image Commun. http://dx.doi.org/10.1016/J.IMAGE.2018.04.015.

[28] G. Toderici, S.M. O'Malley, S.J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, R. Sukthankar, Variable rate image compression with recurrent neural networks, in: Proc. 4th Int. Conf. Learning Representations, 2016..

[29] G. Toderici, D. Vincent, N. Johnston, S.J. Hwang, D. Minnen, J. Shor, M. Covell, Full resolution image compression with recurrent neural networks, in: IEEE Conf. Comput. Vision & Pattern Recognition, 2017, pp. 5435–5443, http://dx.doi.org/10.1109/CVPR.2017.577.

[30] L. Medsker, L. Jain, Recurrent neural networks, Des. Appl., 5, http://dx.doi.org/10.4249/scholarpedia.1888.

[31] N. Johnston, D. Vincent, D. Minnen, M. Covell, S. Singh, T. Chinen, S.J. Hwang, J. Shor, G. Toderici, Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks, arXiv preprint arXiv:1703.10114.

[32] J. Ballé, V. Laparra, E.P. Simoncelli, End-to-end optimized image compression, in: Proc. 5th Int. Conf. Learning Representations, 2017.

[33] L. Theis, W. Shi, A. Cunningham, F. Huszár, Lossy image compression with compressive autoencoders, arXiv preprint arXiv:1703.00395.

[34] E. Agustsson, M. Tschannen, F. Mentzer, R. Timofte, L. Van Gool, Extreme learned image compression with gans, in: Proc. IEEE Conf. Comput. Vision & Pattern Recognition Workshops, vol. 1, 2018, p. 2.

[35] Y. Sun, A. Lu, L. Yu, AHG8: WS-PSNR for 360 video objective quality evaluation, Joint Video Exploration Team of ITU-T SG16 WP3 & ISO/IEC JTC1/SC 29.

[36] C.-Y. Liou, J.-C. Huang, W.-C. Yang, Modeling word perception using the elman network, Neurocomputing 71 (16–18) (2008) 3150–3157, http://dx.doi.org/10.1016/J.NEUCOM.2008.04.030.

[37] C.-Y. Liou, W.-C. Cheng, J.-W. Liou, D.-R. Liou, Autoencoder for words, Neurocomputing 139 (2014) 84–96, http://dx.doi.org/10.1016/J.NEUCOM.2013.09.055.

[38] G. Huang, Z. Liu, K.Q. Weinberger, L. van der Maaten, Densely connected convolutional networks, in: Proc. IEEE Conf. Comput. Vision & Pattern Recognition, vol. 1, 2017, p. 3, http://dx.doi.org/10.1109/CVPR.2017.243.

[39] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, arXiv preprint arXiv:1502.03167.

[40] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, in: Proc. 14th Int. Conf. Artificial Intell. & Stat. 2011, pp. 315–323.

[41] Y. He, X. Xiaoyu, Y. Yan, et al., ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11: 360Lib software manual, arXiv preprint arXiv:1803.05506.

[42] L. Li, Z. Li, M. Budagavi, H. Li, Projection based advanced motion model for cubic mapping for 360-degree video, in: IEEE Int. Conf. Image Process, IEEE, 2017, pp. 1427–1431, http://dx.doi.org/10.1109/ICIP.2017.8296517.

[43] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Proc. 3rd Int. Conf. Learning Representations, 2015.