IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 18, 2019, accepted April 10, 2019, date of publication April 16, 2019, date of current version April 30, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2911403

Deep Image Compression in the Wavelet Transform Domain Based on High Frequency Sub-Band Prediction

CHUXI YANG, YAN ZHAO, (Member, IEEE), AND SHIGANG WANG
College of Communication Engineering, Jilin University, Changchun 130012, China

Corresponding author: Yan Zhao (zhao_y@jlu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61631009, and in part by the National Natural Science Foundation of China under Grant 61771220.

ABSTRACT In this paper, we propose to use deep neural networks for image compression in the wavelet transform domain. When the input image is transformed from the spatial pixel domain to the wavelet transform domain, one low-frequency sub-band (LF sub-band) and three high-frequency sub-bands (HF sub-bands) are generated. Low-frequency sub-band is firstly used to predict each high-frequency sub-band to eliminate redundancy between the sub-bands, after which the sub-bands are fed into different auto-encoders to do the encoding. In order to further improve the compression efficiency, we use a conditional probability model to estimate the context-dependent prior probability of the encoded codes, which can be used for entropy coding. The entire training process is unsupervised, and the auto-encoders and the conditional probability model are trained jointly. The experimental results show that the proposed approach outperforms JPEG, JPEG2000, BPG, and some mainstream neural network-based image compression. Furthermore, it produces better visual quality with clearer details and textures because more high-frequency coefficients can be reserved, thanks to the high-frequency prediction.

INDEX TERMS Image coding, neural networks, discrete wavelet transforms, predictive models.

I. INTRODUCTION

In recent years, Machine Learning (ML) has been widely used in various fields of image processing and achieves the significant performance, such as image recognition [1]–[3], image classification [4], [5], image segmentation [6], [7], *etc.*. In the field of image compression, the auto-encoder [8] is a network that can be trained for dimension reduction [9], which makes it possible for the implementation of ML-based image compression. However, due to the limited fitting ability of the shallow neural networks and the limitation of hardware computing ability, the performance of image compression using shallow auto-encoder was unsatisfactory. Therefore, ML-based image compression was not systematically formed in the early stage. With the proposal of deep neural networks (DNN) [10] and convolutional neural networks (CNN) [11], [12], image compression based on deep neural networks (deep image compression) gradually shows advantages. In 2016, G. Toderici *et al.* [13] proposed an image compression system based on recurrent neural networks (RNN) [14], [15], which was the first time to train a DNN completely as an image compression system. This compression model was used to compress images with size 32×32, which can yield equal quality to JPEG [16] at the same compression ratio. Subsequently, deep image compression became widely concerned by the researchers. At present, deep image compression can be roughly divided into three categories: recurrent auto-encoder based compression [13], [17]–[19], classical auto-encoder by end-to-end training [20]–[24], [41] and conceptual image compression methods [25].

In the same way as the traditional image compression, a deep image compression system can be divided into four major parts: encoder, decoder, quantization, and entropy coding. The main role of the encoder is to reduce the dimension of the input data and make the data more compact for further compression. Quantization is the process to discrete the continuous valued codes, making it possible to represent

The associate editor coordinating the review of this manuscript and approving it for publication was Valentina E. Balas.

each code unit with limited bits, which is a lossy process. Then the quantized codes are input to the decoder, which is used to produce the reconstructed image by decoding process. The entropy coding is used to further compress the codes by lossless compression, where the entropy can be calculated according to the prior probability of the codes.

Different from the traditional image compression, in a deep image compression system, the four major parts are usually composed by neural networks, and the parameters in the compression model are optimized iteratively in the training. There are two challenges to train the compression model: first, How to deal with the non-differentiable quantization in the back propagation; second, How to control the trade-off between distortion and bit rates.

We focus on the second challenge and propose a deep image compression approach in the wavelet transform domain based on high frequency sub-band prediction. The compression model is trained to extract features in wavelet sub-bands, instead of in the spatial pixel domain image. Compared with the image compression in the spatial pixel domain, there are two advantages for our compression approach: firstly, the wavelet transform separates the low-frequency and the high-frequency information of the input image. As a result, each sub-band can be processed respectively according to different characteristics, which is helpful to the reservation of high-frequency information to improve the quality of detail reconstruction; Secondly, after the discrete wavelet transform (DWT) [26], the probability distribution of high-frequency coefficients is relatively concentrated, which is conducive for modeling the prior probability of high-frequency codes. Furthermore, inspired by the idea of wavelet zero tree compression [27], we use neural networks to predict high frequency sub-bands (HF sub-bands) according to low frequency sub-band (LF sub-band), in order to remove the redundancy between LF sub-band and HF sub-bands.

II. RELATED WORKS

A. TRODITIONAL IMAGE COMPRESSION

Currently, JPEG and JPEG2000 [28] are the international standards for image compression, and h. 265/HEVC [29] is the latest video compression standards, which can also implement the intra-frame coding as image compression. JPEG is the earliest proposed image compression standard based on discrete cosine transform (DCT) [30]. After the image is decomposed by DCT, the high-frequency coefficients are dramatically quantized and compressed to realize data compression. However, each coefficient after DCT is related to all the pixels of the whole image; thus, it requires a large amount of computation. Therefore, block segmentation is carried out before implementing the subsequent compression, which leads to some block artifacts [31] under low bit rate. Compared with JPEG, JPEG2000 is based on DWT. Because the length of the base function of the DWT is variable, the coefficients after DWT only reflect some local features of the input image, showing good local characteristics. Thus, the block artifacts can be well avoided in the JPEG2000 because the compression is carried out without block segmentation. BPG [32] is a recently proposed image compression method based on a subset of the HEVC, which has smaller compressed files than JPEG and JPEG2000 for similar quality because the correlation between adjacent blocks are considered and up to 35 prediction modes are adopted in the compression.

In traditional image compression methods, the parameters and compression modes need to be set artificially for transformation, quantization, and coding process. Even for BPG coding with multiple prediction modes, each prediction mode is fixed, which lacks flexibility, and which limits the elimination of redundancy in the compression system. In addition, because the encoding and decoding processes are carried out independently, it is difficult to optimize the encoded codes directly according to the quality of decoded image, so there is still room for optimization in the encoding and decoding process.

B. DEEP IMAGE COMPRESSION

In deep image compression, an auto-encoder based on DNN is trained as encoder and decoder, which maps the input image to codes, and then inversely maps the codes to the output image, with the target of reducing the distortion between the input image and the output image. Data dimension of the code is usually lower than that of the input image, in this way, the reduction of data dimension can be realized. Much attention has been attracted to deep image compression since 2016. Initially, the compression was mainly implemented by a RNN-based auto-encoder with progressive training [13], [17]–[19]. In each recurrence, residual between the latest reconstruction and the input image was encoded by the RNN-based auto-encoder. Encoded bits of each residual are added to the total bits iteratively and finally a progressive-increasing encoded bit stream is formed. However, the value range of the residual, as well as the speed of convergence will change by iteration, making it a challenge to make a trade-off between different residuals in a single auto-encoder. Another kind of deep image compression is implemented by a DNN-based auto-encoder with end-to-end training [20]–[24], [41], in which there is no recurrent structure in both of the encoder and the decoder, thus, encoded bits can be obtained from the encoder all at one time. Compared with the RNN-based image compression, there is no need for the adaption to the progressive-changing residuals in the end-to-end image compression, which brings a more stable training and simpler structure to the auto-encoder.

Currently, almost all the deep image compression approaches are trained for the pixel domain image, making a lack of high frequency sensitivity–some of the details and textures of the image will become over-smoothed due to the loss of high frequency information. Therefore, it is difficult to meet the requirement of high quality HF reconstruction by means of pixel-domain deep image compression.

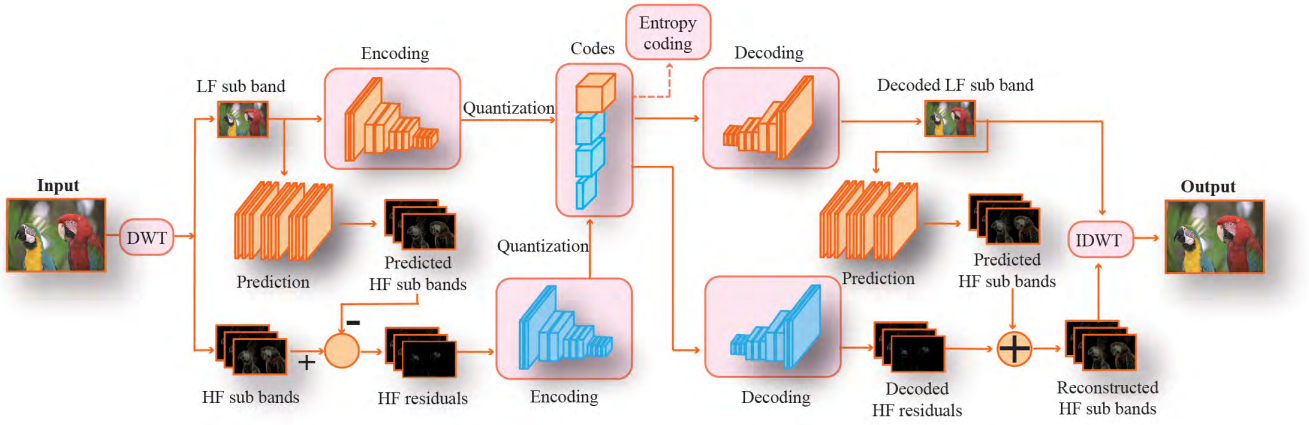


FIGURE 1. Framework of the proposed image compression based on high-frequency sub-band prediction.

III. DEEP IMAGE COMPRESSION IN THE WAVELET TRANSFORM DOMAIN

A. FRAMEWORK

Fig. 1 is the framework of our compression model. The input image x is firstly decomposed into four sub-bands $\{x^{(s)}\}$ by DWT, which produces one LF sub-band and three HF sub-bands. Four parallel auto-encoders are established to encode/decode the sub-bands respectively. After the encoding, each sub-band $x^{(s)}$ is mapped to a latent representation $z^{(s)}$, followed by the quantization (yields the codes $\bar{z}^{(s)}$). Then, the decoders do the inverse mapping and output the reconstructed sub-bands $\{\hat{x}^{(s)}\}$. Finally, the reconstructed image $\hat{x}$ is obtained by inverse discrete wavelet transform (IDWT) from the reconstructed sub-bands. we can compactly represent the networks as follows:

$$\{x^{(s)} = DWT(x) \tag{1}$$
$$z^{(s)} = \mathbb{E}^{(s)}(x^{(s)}) \tag{2}$$
$$\bar{z}^{(s)} = \mathbb{Q}^{(s)}(z^{(s)}) \tag{3}$$
$$\hat{x}^{(s)} = \mathbb{D}^{(s)}(\bar{z}^{(s)}) \tag{4}$$
$$\hat{x} = IDWT(\{\hat{x}^{(s)}\}) \tag{5}$$

where $\{x^{(s)}$ represents four sub-bands after DWT. The superscript (s) is the index of different sub-bands in $\{x^{(s)}\}$. $\mathbb{E}^{(s)}$, $\mathbb{D}^{(s)}$, and $\mathbb{Q}^{(s)}$ represent the encoding, decoding and quantization respectively. $\bar{z}^{(s)}$ is the quantized latent representation (i.e. the codes).

In our compression approach, at the encoder side, HF sub-bands are firstly predicted by LF sub-band before encoding. Thus, in the encoding process, the LF sub-band is directly input into LF encoder, while the HF residuals, instead of HF sub-bands are input into HF encoders because some of the HF information can be predicted by the LF sub-band. The network is then extended as follows:

$$(x^{(ll)}, \{x^{(h)}\}) = DWT\ (x) \tag{6}$$
$$\hat{x}^{(ll)} = \mathbb{D}^{(ll)}(\mathbb{Q}^{(ll)}(\mathbb{E}^{(ll)}(x^{(ll)}))) \tag{7}$$
$$\hat{x}^{(h)} = \mathbb{D}^{(h)}(\mathbb{Q}^{(h)}(\mathbb{E}^{(h)}(r^{(h)}))) + \mathbb{P}^{(h)}(\hat{x}^{(ll)}) \tag{8}$$
$$\hat{x} = IDWT(\hat{x}^{(ll)}, \{\hat{x}^{(h)}\}) \tag{9}$$

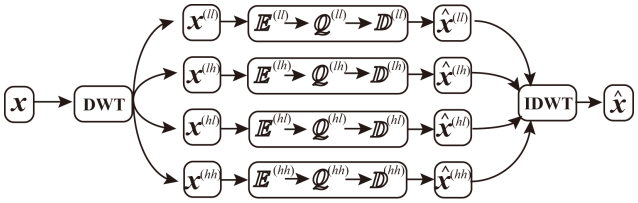


FIGURE 2. Parallel branches of four sub-bands.

where the superscript (ll) is the LF sub-band, (h) is the index of different HF sub-bands. $r^{(h)}$ is the residual between the HF sub-band (h) and its prediction. $\mathbb{P}^{(a)}(b)$ represents the prediction process which to predict sub-band (a) according to sub-band (b).

In order to further improve the compression efficiency, a conditional probability model is adopted to estimate the context-correlated prior probability of $\bar{z}^{(s)}$, which is used in the entropy coding to calculate the entropy. The auto-encoders and the probability model are trained jointly, according to the combined loss function:

$$Loss = Loss_{AE} + Loss_P \tag{10}$$

where $Loss_{AE}$ is the loss of the auto-encoder, and $Loss_P$ is the loss of the probability model.

B. DISCRETE WAVELET TRANSFORMS

Haar wavelet [33] is used in our model to transform the input image from the spatial pixel domain to the wavelet domain. $x^{(ll)}$, $x^{(lh)}$, $x^{(hl)}$, and $x^{(hh)}$ sub-bands are generated after the DWT, where $x^{(ll)}$ is the LF sub-band, $x^{(lh)}$, $x^{(hl)}$, and $x^{(hh)}$ are the HF sub-bands. The LF sub-band contains low frequency information of the input image, which can be seen as an approximation of the input image because most of the information is concentrated in the LF sub-band. Three HF sub-bands contain some high frequency information with horizontal, vertical, and diagonal textures, respectively. The sub-bands are processed along four parallel branches, as shown in Fig. 2.

The end-to-end training is adopted in the whole system, i.e., the branches are not trained separately, instead, four branches are trained jointly from the input-end x to the output-end $\hat{x}$. Compared with separate training, four parallel branches are connected when produced the reconstructed image by IDWT in the joint training. Accordingly, when the model is optimized by back propagation, the correlation between sub-bands can be considered in the process of optimization. Therefore, the latent representation of each sub-band can be optimized directly according to the quality of reconstructed image. In addition, after adding the entropy coding model, the bit allocation between sub-bands can be better optimized in joint training, because the correlation between sub-bands can also be considered in the bitrate optimization.

C. PREDICTION OF HIGH FREQUENCY SUB-BANDS

The LF sub-band can be continually decomposed by the wavelet transform stage by stage, which is referred to as multi-stage wavelet transform. After multi-stage wavelet transform, the sub-bands obtained by the transform present a tower-like distribution, forming a wavelet tree, as shown in Fig. 3.

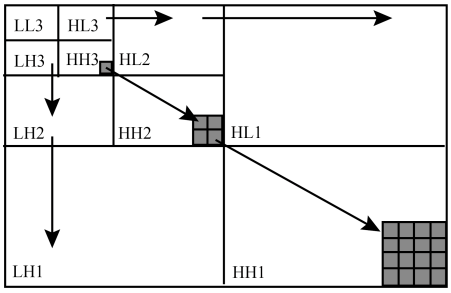


FIGURE 3. Tower-like distribution of sub-bands obtained from multi-stage wavelet transform.

According to the theory of image coding based on wavelet zero-tree, there are strong correlations between the coefficients of the homonymous sub-bands from different stages. For example, in Fig. 3, the coefficients of the three shaded blocks in HH sub-band from level one, two, and three show a strong correlation. As a result, some coefficients can be predicted by the homonymous sub-bands to remove some redundancy.

In our wavelet based compression system, the LF sub-band $x^{(ll)}$ can be continually decomposed by DWT to produce one LF sub-band $x^{(ll_2)}$ and three HF sub-bands $x^{(h_2)}$. Since the wavelet transform is invertible, from the perspective of information theory, there is no loss of information before and after the wavelet transform, i.e.:

$$\begin{aligned} I\left(x^{(ll)}\right) &= I\left(x^{(ll_2)}\right) + I(x^{(h_2)}|x^{(ll_2)}) \\ &= I\left(x^{(h_2)}\right) + I(x^{(ll_2)}|x^{(h_2)}) \end{aligned} \tag{11}$$

where $I(*)$ is the self-information, and $I(*|*)$ is the conditional self-information. According to the wavelet tree theory, $x^{(h)}$ sub-band can be predicted by $x^{(h_2)}$ sub-band because of the strong correlation. And according to (11), since all the information of $x^{(h_2)}$ is included in the self-information of $x^{(ll)}$, $x^{(ll)}$ can be used to make predictions for $x^{(h)}$. Therefore, we add a HF prediction before encoding, and encode only the HF residuals to improve compression efficiency. The residual of each HF sub-band can be represented as:

$$r^{(h)} = x^{(h)} - \mathbb{P}^{(h)}(x^{(ll)}) \tag{12}$$

However, in the lossy compression system, some of the LF information will get loss when transmitting from the encoder to the decoder, we have $I(\hat{x}^{(ll)}) < I(x^{(ll)})$, thus, there will be $I(\mathbb{P}^{(h)}(\hat{x}^{(ll)})) < I(\mathbb{P}^{(h)}(x^{(ll)}))$, i.e., compared with the HF prediction at the encoder side, some information of HF prediction will get loss at the decoder. Unfortunately, this part of loss information is not transmitted to the decoder because it has been subtracted at the encoder, which causes some extra loss of HF information. In order to solve this problem, at the encoder side, when we get the predicted HF sub-bands, an extra network was trained to yield a mask with the same size of the prediction, which is represented by $M^{(h)}$. The range of each value in $M^{(h)}$ is (0, 1), and each HF prediction is masked by $M^{(h)}$ before subtracted by $x^{(h)}$, then (12) becomes:

$$r^{(h)} = x^{(h)} - \mathbb{P}^{(h)}\left(x^{(ll)}\right) \circ M^{(h)} \tag{13}$$

where $\circ$ represents the Hadamard product produced by element multiplication.

Since $M^{(h)}$ can be optimized by the gradient descent according to the gradient with respect to the reconstructed image, which can be seen as a feedback from decoder to the encoder. Therefore, after filtered by the mask, the predictions can be appropriately kept at the encoder according to the quality of reconstructed image.

After the prediction process, the HF residuals obtained according to (13) are input into the HF encoders $E^{(h)}$, while the LF sub-band is directly input into the $E^{(ll)}$. At the decoder side, the HF prediction is also made according to the decoded LF sub-bands $\hat{x}^{(ll)}$. The structure of the prediction network at the decoder is same as the one at the encoder, but without the mask network because all the predictions can be used for the HF reconstruction at the decoder side. The reconstructed HF sub-bands are obtained by the combination of decoded HF residuals and the HF predictions, which can be represented as:

$$\hat{x}^{(h)} = \hat{r}^{(h)} + \mathbb{P}^{(h)}\left(\hat{x}^{(ll)}\right) \tag{14}$$

D. AUTO-ENCODERS FOR DIFFERENT SUB-BANDS

The encoders and decoders of four sub-bands are implemented fully by convolutional neural networks. At the encoder side, convolutional layers with different strides are used to extract multiple-scale features of each sub-band. After the encoding, each encoder yields a latent representation $z^{(s)} \in \mathbb{R}^{w \times h \times n^{(s)}}$, where $n^{(s)}$ is the number of feature maps, w and h are the width and height of each feature map, respectively. At the decoder side, transpose convolutional

layers with different strides are used to reconstruct the corresponding sub-band from $\bar{z}^{(s)}$, which performs the counterpart inverse operations to the encoder.

The size of each feature map ($w \times h$) is same for all the latent representations of four sub-bands, thus, the bit allocation between sub-bands can be roughly controlled by setting different $n^{(s)}$ to each sub-band. In general, since most information of the input image is concentrated in the LF sub-band, $n^{(ll)}$ is set much bigger than $n^{(h)}$, besides, $n^{(hh)}$ is set the smallest of all because $x^{(hh)}$ contains the fewest information of the input image. However, it is hard to precisely adjust the bit allocation by simply set $n^{(s)}$ artificially because the need of $n^{(s)}$ may differ in different images. Therefore, we add a branch to yield an important map [24] at the end of each encoder, which is used as a filter to filter out some codes from the last few feature maps in $z^{(s)}$ by means of setting the corresponding codes to zero. The important map is represented as $Im^{(s)} \in \mathbb{R}^{w \times h \times 1}$, and each value in $Im^{(s)}$ is normalized to (0, 1) by the sigmoid function. The filtering process can be represented as:

$$I(z_{ijk}^{(s)}) = \begin{cases} z_{ijk}^{(s)}, & k/n^{(s)} \leq Im_{ij}^{(s)} \\ 0, & k/n^{(s)} > Im_{ij}^{(s)} \end{cases} \tag{15}$$

where $\mathbb{I}(*)$ represents the filtering process by important map, the subscripts i, j, and k represent the coordinates on the dimension of width, height, and feature map channel, respectively. The important map is trained iteratively by optimizing both reconstructive quality and the bit cost, therefore, the bit allocation between sub-bands can be fine-tuned by the important map according to different kinds of image contents. Furthermore, since an approximate of the important map can be recovered according to $I(z^{(s)})$ by:

$$\hat{Im}_{ij}^{(s)} = \frac{k}{n^{(s)}}, k = \underset{k}{argmax}\left(I(z_{ijk}^{(s)}) \neq 0\right) \tag{16}$$

and it will give the same result after filtered by $\hat{Im}^{(s)}$ as filtered by $Im^{(s)}$. As a result, the important maps are not needed to be coded, nor transmitted to the decoder.

After filtered by the important map, the filtered latent representation $\mathbb{I}(z^{(s)})$ is obtained, then comes the quantization (see section III.E). $\mathbb{I}(\bar{z}^{(s)})$ is the code that actually to be decoded at the decoder, with the bitrate of $R(\mathbb{I}(\bar{z}^{(s)}))$. To better control the trade-off of reconstruction error and the bit rates, a rate-distortion optimization function is used as the loss function of the auto-encoders:

$$Loss_{AE} = \lambda \bullet D\left(x, \hat{x}\right) + \sum\nolimits_s R(\mathbb{I}(\bar{z}^{(s)})) \tag{17}$$

where D denotes the distortion loss between the input image and the output image, R denotes the bitrate loss, and λ is a weight parameter introduced to balance D and R.

E. QUANTIZATION

The non-uniform quantization is adopted in the system, let $\Omega^{(s)} = [\omega_1^{(s)}, \omega_2^{(s)}, \ldots, \omega_q^{(s)}] \in \mathbb{R}^{1 \times q}$ represents the quantization centers of sub-band (s), which are optimized during training, q is the number of quantization centers. The results of quantization are determined by the 2-norm between $I(z^{(s)})$ and $\Omega^{(s)}$, which can be represented as:

$$\begin{aligned} \bar{z}_i^{(s)} &= \mathbb{Q}^{(s)}\left(\mathbb{I}(z^{(s)})_i\right) \\ &= \underset{w_j^s}{argmin} \left\| \mathbb{I}(z^{(s)})_i - \omega_j^{(s)} \right\|_2 \end{aligned} \tag{18}$$

where the subscript i is the index of each code, and j is the index of quantization center.

According to (18), the quantization function is a step function whose derivative at quantization centers is the impact response, and the derivative at other points is zero. Thus, the networks cannot be optimized by gradient descent in back propagation. In order to solve the problem, in the back propagation, we adopt a differentiable approximation of the quantization function, the quantization result is determined by all the quantization centers, which are weighted by normalized 2-norm distance, i.e.:

$$\frac{d}{dz}\bar{z}^{(s)} = \frac{d}{dz}\hat{\mathbb{Q}}\left(\mathbb{I}(z^{(s)})\right) \tag{19}$$

In the forward propagation:

$$\hat{\mathbb{Q}}\left(\mathbb{I}(z^{(s)})\right) = \mathbb{Q}(\mathbb{I}(z^{(s)})) \tag{20}$$

In the back propagation:

$$\hat{\mathbb{Q}}\left(\mathbb{I}\left(z^{(s)}\right)_i\right) = \sum\nolimits_{j=1}^{q} \sigma_j(\left\| \mathbb{I}(z^{(s)})_i - \omega_j^{(s)} \right\|_2) \times \omega_j^{(s)} \tag{21}$$

where $\sigma(*)$ represents the softmax function.

F. ENTROPY CODING

Entropy coding is a lossless coding process based on the principle of information entropy, where the entropy can be used as the expectation of bit cost of the codes. After the quantization, the quantized latent representation of four sub-bands are obtained, which can be represented as $\bar{z}^{(ll)}$, $\bar{z}^{(lh)}$, $\bar{z}^{(hl)}$, and $\bar{z}^{(hh)}$, where $\bar{z}^{(s)} \in \mathbb{R}^{w \times h \times n^{(s)}}$. Let $L^{(s)} = w \times h \times n^{(s)}$ denotes the code length of $\bar{z}^{(s)}$, the entropy of $\bar{z}^{(s)}$ can be represented as:

$$\begin{aligned} H\left(\bar{z}^{(s)}\right) &= E\left(R\left(\bar{z}^{(s)}\right)\right) \\ &= E_{\bar{z}^{(s)} \sim p(\bar{z}^{(s)})}\left[-log_2\left(p\left(\bar{z}^{(s)}\right)\right)\right] \\ &= E_{\bar{z}^{(s)} \sim p(\bar{z}^{(s)})}\left[\sum_i -log_2\left(p\left(\bar{z}_i^{(s)}\right)\right)\right] \end{aligned} \tag{22}$$

where $E\left(\mathrm{R}\left(\bar{z}^{(s)}\right)\right)$ represents the expectation of the bit cost of $\bar{z}^{(s)}$, $i \in \{1, 2, \ldots, L^{(s)}\}$ is the index of each code, $p\left(\bar{z}^{(s)}\right)$ represents the prior probability of $\bar{z}^{(s)}$.

According to (22), since the entropy is directly related to the prior probability of $\bar{z}^{(s)}$, we can transform the problem from solving entropy into solving the prior probability. The prior probability can be simply estimated as a uniform

distribution model, i.e., $p\left(\bar{z}_i^{(s)}\right) = 1/q$, we have $H\left(\bar{z}^{(s)}\right) = \mathbb{E}_{\bar{z}\sim u(1/q)}\left[\sum_i -log_2\left(1/q\right)\right] = L^{(s)} \times log_2\left(q\right)$. This is obviously a compression with redundancy because sometimes, the probability of code is context-dependent, e.g., the probability of a blue color appearing in a blue sky is rather high, which may making smaller bitrate than $L^{(s)} \times log_2\left(q\right)$, therefore, a more reasonable prior probability model should be established to improve the compression efficiency.

We modeled the $p\left(\bar{z}^{(s)}\right)$ by a conditional probability model inspired by [23], where each code $\bar{z}_i^{(s)}$ is generated based on all the codes in front of $\bar{z}_i^{(s)}$:

$$p\left(\bar{z}_i^{(s)}\right) = p\left(\bar{z}_i^{(s)} \mid \bar{z}_{i-1}^{(s)}, \bar{z}_{i-2}^{(s)}, \ldots, \bar{z}_1^{(s)}\right) \tag{23}$$

Thus, the prior probability of $\bar{z}^{(s)}$ is the joint probability:

$$\begin{aligned} p\left(\bar{z}^{(s)}\right) &= p(\bar{z}_L^{(s)}, \bar{z}_{L-1}^{(s)}, \cdots, \bar{z}_i^{(s)}, \cdots \bar{z}_1^{(s)}) \\ &= \prod_{i=1}^{L^{(s)}} p\left(\bar{z}_i^{(s)} \mid \bar{z}_{i-1}^{(s)}, \bar{z}_{i-2}^{(s)}, \ldots, \bar{z}_1^{(s)}\right) \end{aligned} \tag{24}$$

However, since the number of correlated codes increases as the index i sliding from 1 to $L^{(s)}$, it is hard to establish the probability model because the size of correlated region is unfixed, furthermore, it takes large amount of calculations to consider all the codes in front of $\bar{z}_i^{(s)}$. As a result, only a fixed size of neighborhood of $\bar{z}_i^{(s)}$ is taken into consideration, as shown in Fig. 4.

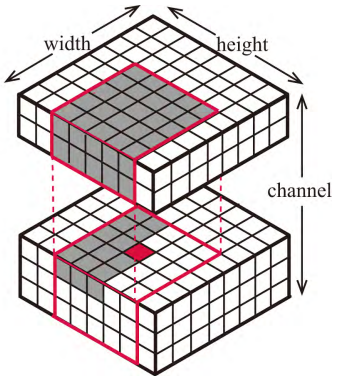


FIGURE 4. The neighborhood of $\bar{z}_i^{(s)}$ being considered in the probability model. The red block represents the center code $\bar{z}_i^{(s)}$, the cube which highlighted by red lines is the neighborhood of $\bar{z}_i^{(s)}$ with the size of $m \times m \times m$ (in this figure $m = 5$). When estimate the probability of $\bar{z}_i^{(s)}$, only the forward codes of $\bar{z}_i^{(s)}$ in the neighborhood region are taken into consideration, which are marked in grey in the figure.

The probability model is implemented by CNN. Since both width, height and channel dimensions are involved in the 3D-neighbourhood, the 3D convolution kernels are used to extract the feature from the 3D region. Since only the forward codes of $\bar{z}_i^{(s)}$ are considered, we use a kernel mask in each convolutional layer to make the convolution kernel "ignore" the code after $\bar{z}_i^{(s)}$. The values in the mask are set as follows: the values after the mask center are set to 0; the values in front of the mask center are set to 1; the value of the mask center is set to 0 for the mask of first layer, whereas is set to 1 for the masks of other layers (the values are set in a raster scan order: from width to height to channel).

The output of the probability model is $p\left(\bar{z}^{(s)}\right) \in \mathbb{R}^{w \times h \times n^{(s)} \times q}$, which corresponds to the probability of $\bar{z}_i^{(s)}$ taking different values of quantization centers in $\Omega^{(s)}$. In order to get a lower bitrate in the entropy coding, the results in (22) is expected as small as possible, i.e., the probability model is trained to achieve the maximum likelihood. We use cross entropy as the loss function of the probability model, assuming p_r is the real probability, and p is the output of the probability model, the cross entropy loss of probability model can be represented by:

$$\begin{aligned} Loss_P^{(s)} &= -\sum\nolimits_i \sum_{j=1}^{q} p_r(\bar{z}_i^{(s)} = \omega_j^{(s)}) log(p(\bar{z}_i^{(s)} = \omega_j^{(s)})) \\ &= -\sum\nolimits_i p_r(\bar{z}_i^{(s)}) log(p(\bar{z}_i^{(s)})) \\ &= -\sum\nolimits_i log(p(\bar{z}_i^{(s)})) \end{aligned} \tag{25}$$

compared with (22) and (25), we have:

$$\begin{aligned} H\left(\bar{z}^{(s)}\right) &= E_{\bar{z}^{(s)}\sim p(\bar{z}^{(s)})}\left[R\left(\bar{z}^{(s)}\right)\right] = E_{\bar{z}^{(s)}\sim p(\bar{z}^{(s)})}\left[Loss_P^{(s)}\right] \\ &\rightarrow R\left(\bar{z}^{(s)}\right) = Loss_P^{(s)} \end{aligned} \tag{26}$$

According to (26), when the probability model is optimized according to the loss function of cross entropy, it can be regarded as a process of directly optimizing the bit cost of $\bar{z}^{(s)}$, which has the benefit to directly minimize the bit cost in the loss function. Combining with (17) and (25), the loss function of the whole system can be represented as:

$$\begin{aligned} Loss &= Loss_{AE} + Loss_P \\ &= \lambda \bullet D\left(\hat{x}, x\right) + \sum\nolimits_s R(\mathbb{I}(\bar{z}^{(s)})) + \sum\nolimits_s R\left(\bar{z}^{(s)}\right) \\ &= \lambda \bullet D\left(\hat{x}, x\right) + \sum\nolimits_s \mathbb{I}(R(\bar{z}^{(s)})) + \sum\nolimits_s R\left(\bar{z}^{(s)}\right) \end{aligned} \tag{27}$$

Note that the $R(\mathbb{I}(\bar{z}^{(s)}))$ is different from $R\left(\bar{z}^{(s)}\right)$ in (27). $R(\mathbb{I}(\bar{z}^{(s)}))$ in $Loss_{AE}$ is the bit cost of codes that actually transmitted to the decoder after masked by the important map, which is optimized according to rate-distortion trade-off. While $R\left(\bar{z}^{(s)}\right)$ in $Loss_P$ is the cross entropy loss of the probability model, which is optimized according to the maximum likelihood, it is numerically equal to the bit cost of all the codes in $\bar{z}^{(s)}$, including the zero codes which has been "filtered-out" by the important map. As a result, $R(\mathbb{I}(\bar{z}^{(s)})) = \mathbb{I}(R(\bar{z}^{(s)})) \leq R\left(\bar{z}^{(s)}\right)$, and $R\left(\bar{z}^{(s)}\right)$ is the bit cost after compression.

IV. NEURAL NETWORK ARCHITECTURES

The proposed compression system consists of wavelet model, prediction model, auto-encoder, and probability model,

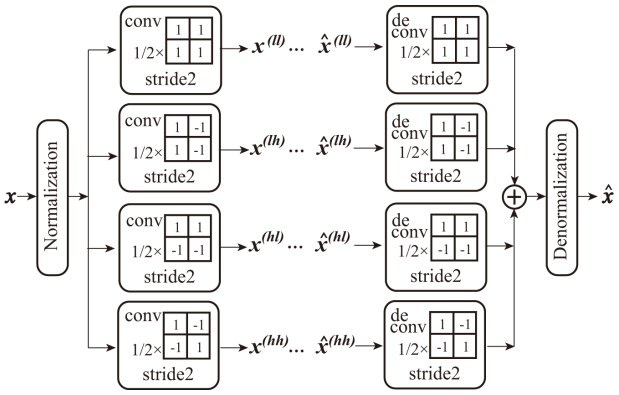


FIGURE 5. The convolution implementation of Haar wavelet model. The input image x is firstly normalized before DWT, and de-normalized after IDWT to yield the output image $\hat{x}$. The "conv" and "deconv" represent convolution and transpose convolution, respectively. "stride2" represents the stride of the sliding window of each dimension is 2.

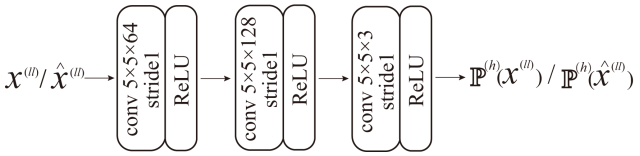


FIGURE 6. The HF prediction model. It is a network with three CNN layers to yield the HF prediction, and the three HF sub-bands are predicted respectively in three parallel branches. The layer "conv 5 × 5 × 64 stride1 ReLU" represents a convolution layer with kernel size 5 × 5, 64 output channels and a stride of 1, followed the by the ReLU activation function.

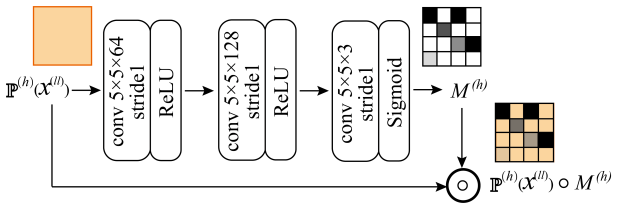


FIGURE 7. The architecture of mask network.

which are all formed by CNN layers. The wavelet model is described in Fig. 5, where four convolution kernels with the size of 2 × 2 are adopted as the wavelet filters of Haar wavelet:

The prediction model of the HF sub-bands and the mask network are described in Fig.6 and Fig.7, respectively:

After the HF prediction, the HF residuals are obtained according to (13), LF sub-band and three HF residuals are input into the auto-encoders, which is described in Fig. 8. The architecture of the probability model is shown in Fig. 9.

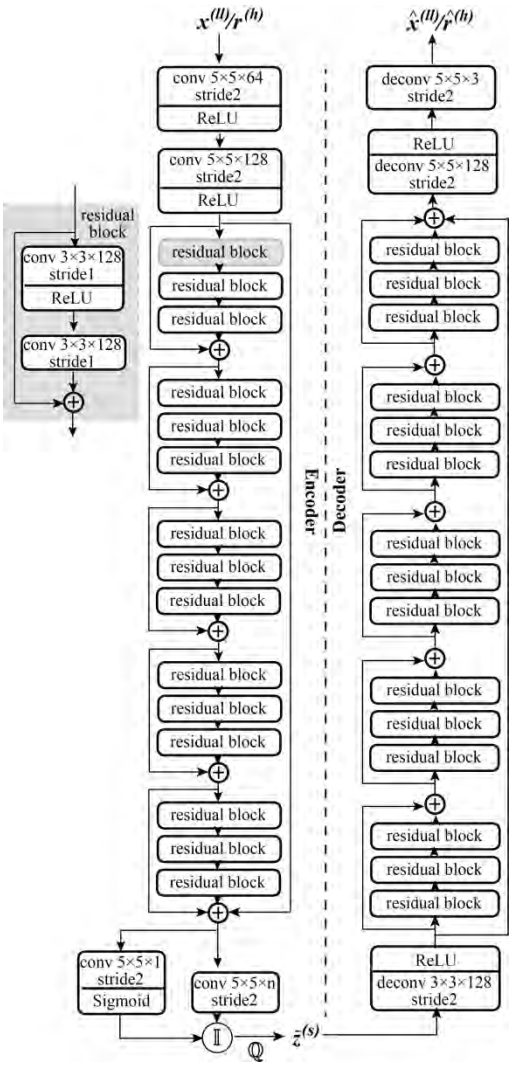


FIGURE 8. The architecture of auto-encoder. The layer "conv/deconv 5 × 5 × 64" represents a convolution/transpose-convolution layer with 64 output channels, and with kernel size 5 × 5. n is the number of channels of each $\bar{z}^{(s)}$, which is set different for different sub-bands (detailed in section V.E). "SAME" padding is used in all the convolution and transpose convolution layers with the padding value of 0.

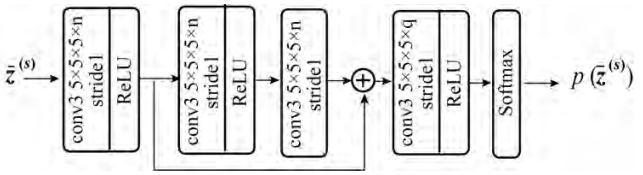


FIGURE 9. The architecture of probability model. "conv3 5 × 5 × 5×n" represents a convolution layer with the 3D-kernel size 5 × 5 × 5, n output channels, where n is the channel number of $\bar{z}^{(s)}$. "SAME" padding is also used in all the convolution layers with the padding value of $\omega_1^{(s)}$.

V. EXPERIMENTS

A. DATASETS

The proposed compression models are trained on a subset of ImageNet [34] with about 440000 3-channel images in RGB color space. Each image in the training dataset is reshaped to 256 × 256 to save the storage. After training, the models are tested on the Kodak PhotoCD image dataset [35].

B. PREPROCESSING FOR IMAGES

The images in the training set are randomly cropped into 160 × 160 patches, and randomly flipped before input into the compression model. In the normalization process, we normalize each channel of the input patch by z-score normalization, according to the mean and variance obtained from the

training set, the normalization can be described as:

$$\mathbb{N}(x_c) = \frac{x_c - mean_c}{\sqrt[2]{var_c}} \tag{28}$$

where the subscript c is the index of different channels of the input patch. At the decoder side, accordingly, the de-normalization is carried out to the image obtained by IDWT, and produce the final reconstructed image.

C. MEASUREMENT OF DISTORTION

In (27), $D\left(x, \hat{x}\right)$ represents the distortion between the input image and the reconstructed image. In deep compression

system, mean squared error (MSE) and multi-scale structural similarity index (MS-SSIM) [36] are generally used as the measurement of distortion. Compared with MSE, MS-SSIM is the quality evaluation metric which can better reflect the image quality observed by human eyes, as a result, we adopt MS-SSIM as the measurement of distortion in training and testing process. During the training, the distortion $D\left(x, \hat{x}\right) = 1 - \text{MS\_SSIM}\left(x, \hat{x}\right)$, which is to be minimized according to the loss function described in (27).

D. TRAINING PARAMETERS

We adopt Adam optimizer [37] to train the models with a mini-batch size of 25. The loss function of the system is (27), which combines the loss of auto-encoder and the loss of probability model to yield a joint training. The initial learning rate of the auto-encoder and probability model are set to 2×10^{-4} and 1×10^{-4}, respectively, with a decay factor of 10 for every 2 epoch. In order to realize a more precise control of the bit cost in the training, we set a bitrate threshold R_t to penalize the bitrate loss only when it is higher than R_t. Therefore, (27) can be re-described by:

$$Loss = \lambda \bullet (1 - MS\_SSIM\left(x, \hat{x}\right)) + max(R - R_t, 0) \tag{29}$$

where $R = \frac{1}{\sum_s L^{(s)}} \sum_s \left[\mathbb{I}(R(\bar{z}^{(s)})) + R\left(\bar{z}^{(s)}\right)\right]/2$, which can be regarded as the average bitrate loss of each code unit. The weight parameter λ is set to 100. In this way, we trained 6 models with different bitrates corresponding to 0.2, 0.3, 0.4, 0.5, 0.6, and 0.7 bits per pixel (bpps), respectively.

E. CHANNEL ALLOCATION BETWEEN SUB-BANDS

The bit allocation of sub-bands can be self-adaptive after adding the important map to $\bar{z}^{(s)}$. Since LF sub-band contains the vast majority of energy of the input image, the value of MS-SSIM is dominantly decided by the reconstructive quality of LF sub-band. As a result, we found that the codes in $\bar{z}^{(ll)}$ tends to be kept preferentially when the model is optimized according to MS-SSIM distortion measure. However, in this situation, the details in the reconstructed image tend to become over-smoothed because of the loss of HF information. Taking the 0.6bpp model as an example, as shown in Fig.10, when the total number of the channels is fixed, comparing the reconstructed images under different channel allocations, it shows that the reconstructed image of

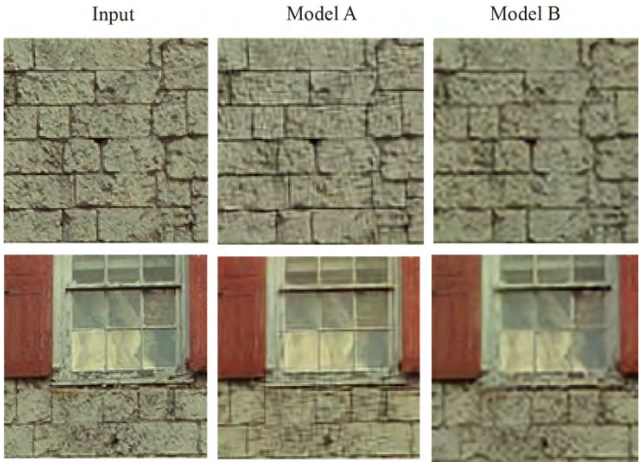


FIGURE 10. Reconstruction results of two models trained under 0.6 bpp. Channel allocation of model A is 64,16,16, and 4 for $\bar{z}^{(ll)}$, $\bar{z}^{(lh)}$, $\bar{z}^{(hl)}$, and $\bar{z}^{(hh)}$, respectively. While in model B the corresponding allocation is 80, 8, 8, and 4. The actual bit cost of LF sub-band and HF sub-bands for the two models are, model A: 26.35kb(LF) / 8.86kb(HF)–MS-SSIM 0.978, model B: 30.03kb(LF) / 6.67kb(HF)–MS-SSIM 0.98, respectively.

TABLE 1. Channel allocation for different models.

bpp	$n^{(ll)}$	$n^{(lh)}$	$n^{(hl)}$	$n^{(hh)}$
0. 2	28	8	8	4
0. 3	36	12	12	4
0. 4	44	12	12	4
0. 5	56	12	12	4
0. 6	64	16	16	4
0. 7	72	20	20	8

"model A" ($n^{(ll)} = 64$) has sharper edges and clearer textures than that of "model B" ($n^{(ll)} = 80$), showing an advantages of decreasing some channels of LF sub-band, in spite of a lower MS-SSIM. Therefore, in order to yield a better reconstruction of details and textures, we artificially reduce some channel of $\bar{z}^{(ll)}$ to guarantee the bits for HF sub-bands. The channel allocations for each model are described in Table 1.

In this way, our compression model makes it possible to reconstructed more HF details by flexibly control the bit allocation of sub-bands, which cannot be achieved in the compression model trained in the spatial pixel domain.

F. OTHER SETTINGS

In the quantization, the number q is set to 6, and the quantization centers are initialized by uniform initialization in the range of $(-2, 2)$. Batch normalization [38] is added after each convolutional and transpose convolutional layers in the auto-encoders. We train and test all the models on a GeForce GTX 1070 GPU, and each model is trained for 6 epochs, which takes around 30 hours.

G. EXPERIMENTAL RESULTS AND COMPARISON

Fig. 11 shows the rate-distortion (R-D) results of different compression methods tested on the Kodak PhotoCD image dataset. We firstly compare our compression approach

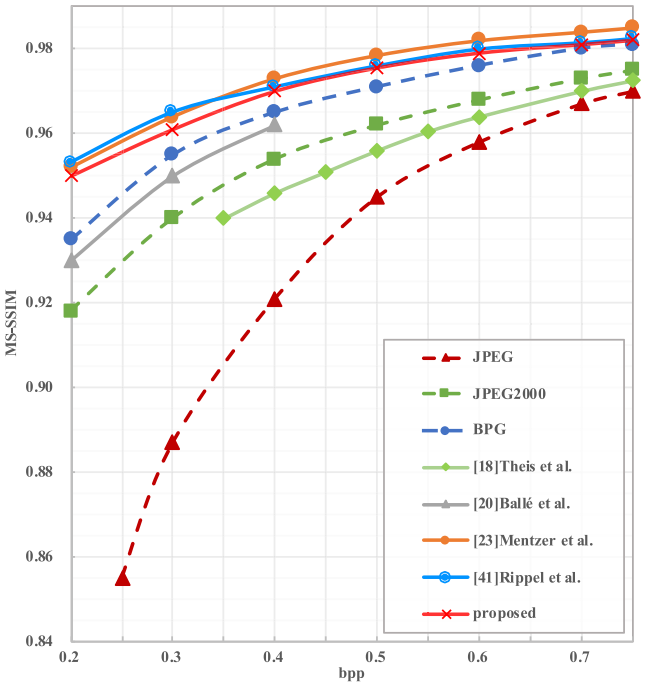


FIGURE 11. Comparison of the ratio-distortion curves by different compression methods.

with some traditional image compression methods, including JPEG, JPEG2000 and BPG, where the JPEG results are generated using the IrfanView software [39]; the JPEG2000 results are generated using JPEG200 Toolbox for matlab contributed by Nikola Sprljan [40]; and the BPG results are generated by the online BPG codec [32]. We use our models to compress the input image, and record the bpp and MS-SSIM values, respectively. Then the target bpps of all the comparison methods are set near to the bpps of our method, and the corresponding MS-SSIM values are collected to form the R-D curves. We also compare our compression approach to some state-of-the-art deep compression methods [18], [20], [23], [41]. When comparing with [23] and [41], we train 6 models for both [23] and [41] on our training set, corresponding to the target bpps shown in Table 1. The implementation of [23] is provided by the authors, and the experimental results of [23] are verified matching to those in the paper. For a fair comparison, when training the models of [23], the number of codes of $\bar{z}^{(s)}$ are set equal to that of our models to carry the same amount of information. Note that, in Fig. 11, the bpp values of the proposed method and of [23] approach are both obtained according to the estimated bit rate values of the coding models. We implement the compression approach as described in [41], and obtain similar results to that shown in [41]. In practice, the scale parameter of [41] is set to 4 for all the models, which is same as the scale of our models. When comparing with [18] and [20], since we have no access to replicate the models described in the papers, we collect the corresponding results test on Kodak dataset from the papers, and seriously transcribe them on our R-D curves.

It can be seen from Fig. 11 that when comparing with the traditional compression methods, the proposed compression approach outperforms JPEG, JPEG2000 and BPG in terms of MS-SSIM, especially under low bit rates. When comparing with the deep compression approaches, the proposed compression approach outperforms Ballé *et al.* [20] and Theis *et al.* [18]. Comparing with [20] and [18], up to 25% and 56% bitrates savings can be achieved respectively using the proposed approach, under the same reconstructive quality. When comparing under the same bpp, the MS-SSIM value of the proposed method is about 0.003 lower than those of the method in [23], which is trained in the spatial pixel domain and shows the outperformed performance in terms of MS-SSIM. Since the adversarial training is used in [41], the compression approach of [41] shows the best performance under low bitrates; whereas when the bitrates is higher than 0.4bpp, the proposed method shows a comparable performance with the compression approach of [41], in terms of MS-SSIM.

We then compare the quality of only HF images between the proposed approach, the methods of [23], and of [41]. The HF images are obtained by IDWT only from HF sub-bands (the coefficients of LF sub-band are set to zeros). Since the models are trained in the pixel domain in both [23] and [41], we first decompose the reconstructed images of [23] and [41] by DWT, then obtain the HF images by the same way. We use MS-SSIM to measure the distortion of HF images. The comparison results under low, middle, and high bitrates are shown in Table 2.

In Table. 2, we highlight the better results with higher MS-SSIM values and lower bitrates at the same time. Compared with Mentzer *et al.* [23], our MS-SSIM of the HF image can averagely achieve 0.0010, 0.0026, and 0.0048 gains under low, middle, and high bitrates, respectively. In particular, for images with more details and textures (e.g. kodim5, kodim8, and kodim13), the HF image of the proposed approach shows outstanding performance in terms of MS-SSIM. Compared with Rippel *et al.* [41], our MS-SSIM of the HF image can averagely achieve 0.0030 and 0.0046 gains under middle and high bitrates, respectively. In particular, our MS-SSIM of the HF image can slightly exceed that of [41] under low bitrates, showing a better reconstruction of high frequency information. Furthermore, it shows in Table 2 that when the total bitrates increase, in the methods of [23] and of [41], the MS-SSIM values of HF images present a slightly increasing, whereas in the proposed method, those of HF images increase obviously, showing an obviously improvement of the detail reconstruction.

Fig. 12 and Fig. 13 shows the visual comparison of some details between different methods under low bitrates and high bitrates, respectively. The results of the proposed approach, of Mentzer *et al.* [23] and of Rippel *et al.* [41] are obtained from the corresponding models trained under 0.3bpp and 0.7bpp.

As shown in Fig. 12, block artifacts occurs obviously in the reconstructed image of JPEG, leading a poor visual

TABLE 2. MS-SSIM comparison between the reconstructed HF images of the proposed approach, of [23] approach, and of [41] approach.

	proposed		Mentzer *et al.* [23]		Rippel *et al.* [41]		proposed		Mentzer *et al.* [23]		Rippel *et al.* [41]		proposed		Mentzer *et al.* [23]		Rippel *et al.* [41]	
Images	bpps	MS-SSIM	bpps	MS-SSIM	bpps	MS-SSIM	bpps	MS-SSIM	bpps	MS-SSIM	bpps	MS-SSIM	bpps	MS-SSIM	bpps	MS-SSIM	bpps	MS-SSIM
kodim01	0.34	**0.9785**	0.36	0.9741	0.36	0.9757	0.59	**0.9853**	0.63	0.9767	0.59	0.9770	0.85	0.9899	0.82	0.9774	0.86	0.9775
kodim02	0.31	**0.9908**	0.31	0.9901	0.33	0.9903	0.53	**0.9923**	0.57	0.9908	0.56	0.9910	0.71	**0.9944**	0.73	0.9911	0.76	0.9911
kodim03	0.30	0.9945	0.28	0.9941	0.33	0.9943	0.48	**0.9955**	0.48	0.9947	0.52	0.9946	0.61	**0.9964**	0.67	0.9949	0.75	0.9951
kodim04	0.32	0.9917	0.31	0.9915	0.34	0.9915	0.52	**0.9927**	0.54	0.9925	0.55	0.9923	0.70	**0.9945**	0.74	0.9927	0.78	0.9926
kodim05	0.36	**0.9835**	0.38	0.9829	0.37	0.9831	0.60	**0.9883**	0.66	0.9859	0.60	0.9852	0.82	**0.9911**	0.87	0.9868	0.86	0.9871
kodim06	0.33	**0.9831**	0.33	0.9811	0.34	0.9813	0.56	**0.9899**	0.59	0.9824	0.56	0.9822	0.76	**0.9920**	0.76	0.9828	0.80	0.9834
kodim07	0.30	0.9952	0.30	0.9951	0.35	0.9953	0.49	0.9956	0.5	0.9959	0.55	0.9961	0.62	**0.9968**	0.69	0.9961	0.77	0.9963
kodim08	0.35	**0.9789**	0.36	0.9751	0.36	0.9770	0.60	0.9843	0.63	0.9771	0.59	0.9772	0.86	0.9912	0.83	0.9778	0.84	0.9792
kodim09	0.29	0.9935	0.27	0.9931	0.32	0.9934	0.45	0.9937	0.47	0.9937	0.52	0.9940	0.60	**0.9960**	0.67	0.9939	0.72	0.9940
kodim10	0.32	0.9935	0.30	0.9928	0.34	0.9936	0.51	**0.9944**	0.52	0.9937	0.55	0.9944	0.67	**0.9960**	0.72	0.9940	0.77	0.9947
kodim11	0.32	**0.9871**	0.33	0.9856	0.35	0.9862	0.55	**0.9908**	0.57	0.9868	0.57	0.9854	0.75	**0.9932**	0.75	0.9872	0.80	0.9877
kodim12	0.30	**0.9925**	0.30	0.9918	0.33	0.9919	0.51	**0.9946**	0.53	0.9923	0.53	0.9925	0.66	**0.9956**	0.70	0.9924	0.73	0.9926
kodim13	0.36	**0.9685**	0.38	0.9675	0.36	0.9666	0.62	0.9783	0.67	0.9704	0.60	0.9660	0.94	0.9841	0.88	0.9709	0.86	0.9693
kodim14	0.35	**0.9848**	0.36	0.9844	0.36	0.9842	0.61	0.9895	0.64	0.9862	0.58	0.9857	0.82	**0.9913**	0.84	0.9866	0.83	0.9867
kodim15	0.29	0.9923	0.28	0.9921	0.31	0.9921	0.47	0.9926	0.49	0.9929	0.52	0.9928	0.62	**0.9945**	0.66	0.9931	0.73	0.9929
kodim16	0.31	**0.9886**	0.31	0.9871	0.34	0.9874	0.54	**0.9928**	0.55	0.9879	0.56	0.9881	0.71	**0.9944**	0.73	0.9881	0.78	0.9883
kodim17	0.33	0.9930	0.31	0.9929	0.33	0.9932	0.52	0.9936	0.54	0.9937	0.55	0.9940	0.68	**0.9953**	0.74	0.9940	0.77	0.9942
kodim18	0.34	**0.9850**	0.35	0.9848	0.35	0.9844	0.58	**0.9886**	0.62	0.9867	0.58	0.9861	0.80	**0.9912**	0.82	0.9871	0.82	0.9865
kodim19	0.33	0.9881	0.32	0.9864	0.35	0.9867	0.54	**0.9906**	0.56	0.9876	0.57	0.9881	0.75	**0.9937**	0.76	0.9879	0.79	0.9884
kodim20	0.25	0.9929	0.23	0.9925	0.29	0.9929	0.41	**0.9936**	0.41	0.9932	0.49	0.9927	0.55	**0.9954**	0.56	0.9933	0.69	0.9935
kodim21	0.30	**0.9879**	0.30	0.9871	0.34	0.9872	0.52	**0.9912**	0.54	0.9883	0.57	0.9856	0.70	**0.9933**	0.74	0.9886	0.77	0.9886
kodim22	0.33	**0.9879**	0.34	0.9874	0.35	0.9874	0.55	**0.9900**	0.60	0.9885	0.57	0.9885	0.77	**0.9930**	0.78	0.9889	0.78	0.9887
kodim23	0.30	0.9959	0.27	0.9962	0.31	0.9963	0.45	0.9959	0.45	0.9967	0.51	0.9968	0.59	**0.9973**	0.65	0.9968	0.75	0.9969
kodim24	0.34	**0.9855**	0.35	0.9850	0.35	0.9854	0.57	**0.9887**	0.60	0.9867	0.58	0.9867	0.80	**0.9924**	0.79	0.9873	0.82	0.9871
Ave.	0.320	**0.9881**	0.318	0.9871	0.340	0.9873	0.532	**0.9910**	0.557	0.9884	0.557	0.9880	0.723	**0.9935**	0.745	0.9887	0.785	0.9889

quality and the disappearance of almost all the details. Blurry distortion will occur in the reconstructed image of JPEG2000, especially in the background region, which is caused by the heavily compression on HF coefficients. The reconstructed image of BPG has the sharpest edges of all the comparison methods. When evaluating the visual performance of the whole image, BPG image shows a clearer and smoother visual quality compared with the reconstructed image of JPEG and of JPEG2000. However, in some local regions with rich high frequency textures, the reconstructed image of BPG turns to become blurry. Whereas in these regions, JPEG2000 may better reconstruct the high frequency textures thanks to the DWT. The reconstructed image of [23] has a balance between foreground and background, which shows an acceptable visual quality but lacks some details. The reconstructed image of [41] shows a satisfactory visual quality with clear edges, however, color distortion may occur under low bitrates. Compared with these methods, our reconstructed images shows better visual results with sharp edge, satisfactory details and clear background. As shown in Fig. 12(b), the textures of the roof can be well reconstructed using our approach, whereas the textures will become blurry or only be partly reconstructed using other methods.

As shown in Fig. 13, when the bitrates increase, visual qualities of JPEG and JPEG2000 improve obviously, however, some block artifacts still exist in the reconstructed image of JPEG. Since DWT is adopted in JPEG2000, it shows local characteristic in the reconstructed image, where the textures in the regions of interest (ROIs) are well reconstructed, whereas the textures in other regions are tend to be ignored. As a result, the quality difference between the ROIs and other regions may have some unpleasant effect

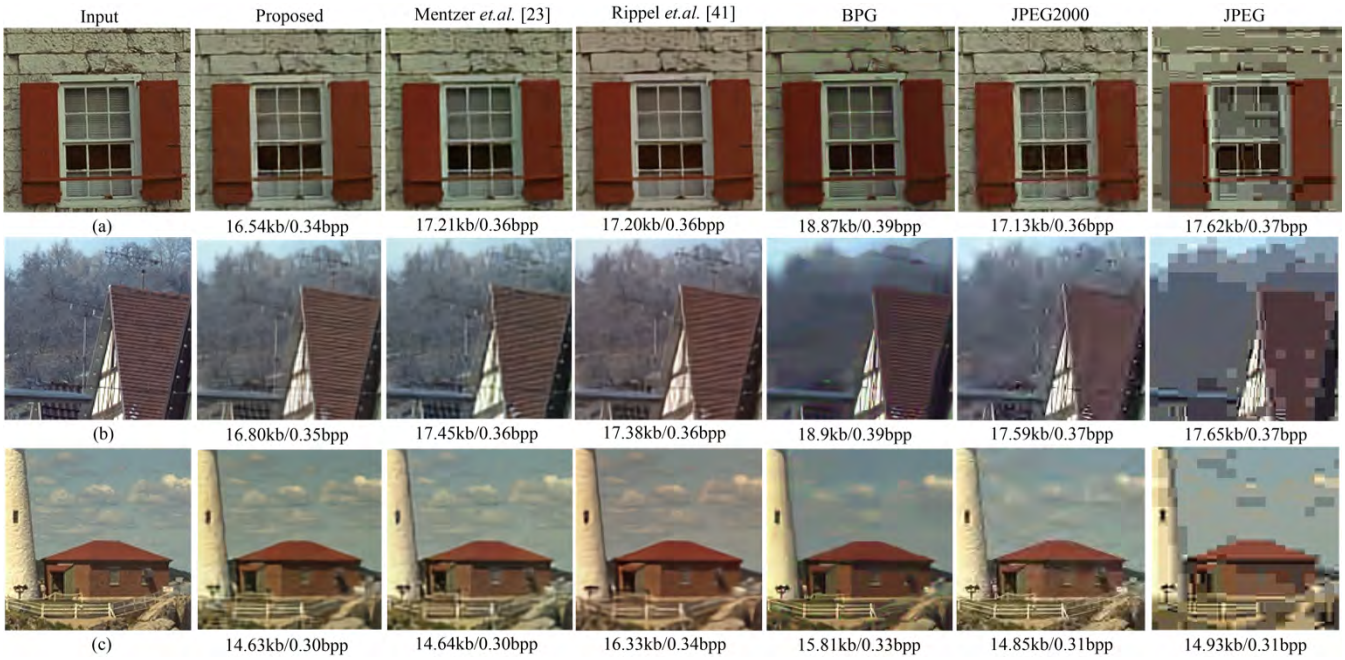


FIGURE 12. Visual comparison between different compression methods under low bitrates. (a), (b), and (c) are the details sampled from *kodim1*, *kodim8*, and *kodim21* from Kodak PhotoCD image dataset.

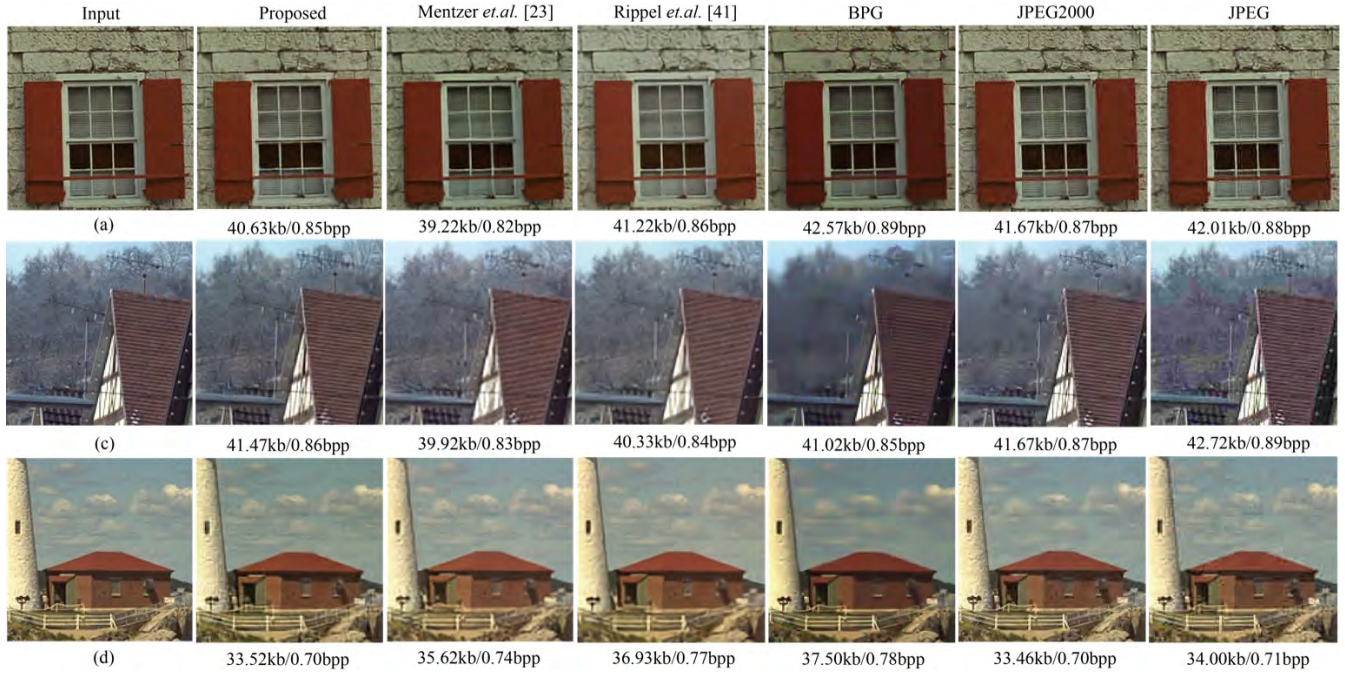


FIGURE 13. Visual comparison between different compression methods under high bitrates.

on the visual quality. In contrast, the textures are evenly reconstructed in the reconstructed image of our method, both in ROIs and other regions, showing a pleasant visual quality. Visual qualities of [23] and [41] are also improved with the increasing of bitrates, however, there are still some high frequency textures which cannot be well reconstructed, showing some high frequency distortions. And the high frequency distortions are hard to be eliminated by simply increasing the bitrates when the models are trained in the pixel domain. Compared with [23] and [41], since our compression models are trained in the wavelet domain, it is easier to reconstruct the high frequency textures because the features

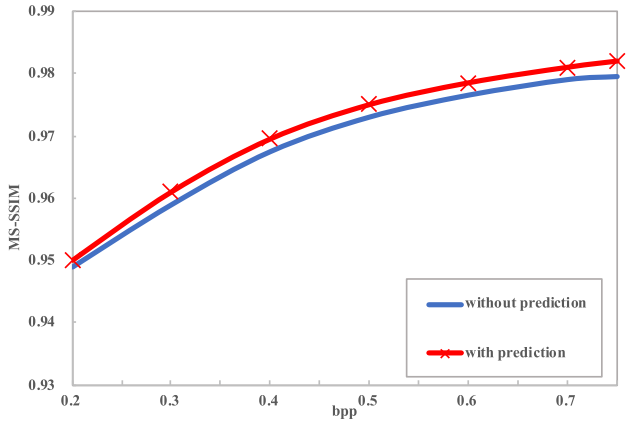


FIGURE 14. R-D comparison between models with and without HF prediction.

TABLE 3. comparison of HF reconstruction between models with and without prediction under 0.6bpp.

Images	with prediction			without prediction		
	bits (kb)	HF bits (kb)	MS-SSIM	bits (kb)	HF bits (kb)	MS-SSIM
kodim01	35.21	8.86	**0.9884**	36.35	12.08	0.9882
kodim02	29.64	5.55	**0.9940**	28.35	6.09	0.9928
kodim03	26.92	3.86	**0.9963**	26.09	4.68	0.9948
kodim04	29.40	4.50	**0.9941**	29.16	6.27	0.9920
kodim05	33.80	5.95	**0.9899**	35.79	10.11	0.9840
kodim06	32.59	7.68	**0.9916**	31.83	8.98	0.9902
kodim07	28.41	4.04	**0.9965**	27.75	5.45	0.9938
kodim08	35.34	8.05	**0.9903**	36.25	11.29	0.9893
kodim09	26.21	3.58	**0.9958**	25.76	5.02	0.9946
kodim10	28.65	4.06	**0.9957**	28.66	6.11	0.9944
kodim11	31.97	7.20	**0.9928**	31.30	8.63	0.9913
kodim12	29.01	5.26	**0.9955**	27.37	5.70	0.9943
kodim13	37.44	9.79	**0.9836**	38.52	13.03	0.9797
kodim14	34.67	7.68	**0.9908**	35.10	10.21	0.9868
kodim15	26.46	3.92	**0.9943**	25.75	4.84	0.9928
kodim16	30.95	6.33	**0.9941**	29.78	7.11	0.9929
kodim17	29.15	3.68	**0.9949**	28.65	5.20	0.9921
kodim18	32.86	6.36	**0.9905**	34.02	9.64	0.9875
kodim19	31.50	6.01	**0.9934**	31.06	7.71	0.9920
kodim20	23.84	4.07	**0.9952**	23.45	5.58	0.9941
kodim21	30.25	6.20	**0.9929**	29.80	7.69	0.9911
kodim22	31.81	5.94	**0.9927**	32.37	8.54	0.9906
kodim23	26.01	2.97	**0.9971**	24.99	3.76	0.9957
kodim24	32.61	6.09	**0.9918**	33.75	9.36	0.9892
Ave.	30.612	5.735	**0.9930**	30.496	7.628	0.9910

of LF and HF sub bands are extracted separately. As a result, the reconstructed images of the proposed method may have richer and clearer textures.

FIGURE 15. Visual comparison between models with and without prediction. Both of the models are trained under 0.6bpp, (a), (b), and (c) are the details sampled from ***kodim7***, ***kodim10***, and ***kodim14*** from Kodak PhotoCD image dataset.

H. COMPRESSION WITHOUT HF PREDICTION

In order to show the effectiveness of the HF prediction, we remove the prediction steps in both encoder and decoder, i.e., all the sub-bands are directly input into the auto-encoders, and other settings are all same as those of the model with prediction. Six models without prediction are trained, and the rate-distortion comparison is shown in Fig. 14. Experimental results show that adding the predictive model will improve the MS-SSIM value by an average of 0.0019. Taking the model trained under 0.6bpp as an example, we evaluate the bit cost of HF sub-bands and the quality of HF image at the same time, and the results are shown in Table. 3. In Table. 3, the total bit cost (kbyte), the bit cost of HF sub-bands (kbyte), and the MS-SSIM of HF image are all listed, which can be seen that after adding the prediction model, the bit cost of HF sub-bands is highly reduced by an average of 24.8%, and the MS-SSIM value of the HF image is increased by an average of 0.002. It means that the prediction model can effectively predict the HF information according to LF sub-band, which is beneficial to produce a better high frequency reconstruction using lower HF bit cost.

Visual comparison of some details are shown in Fig.15. From Fig. 15 we can see that there are some jagged edges in the reconstructed image obtained by the model without prediction, whereas the edges are smoother in the reconstructed image obtained by the model with prediction. In addition, after adding the prediction module, the details and textures of the reconstructed image become clearer.

Above all, the addition of the prediction model can effectively remove the redundancy of the HF sub-bands, furthermore, visual quality of the reconstructed image can be improved obviously.

VI. CONCLUSIONS & FUTURE WORKS

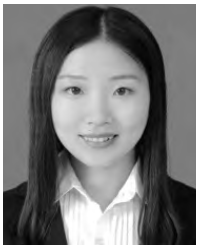
In this paper, we proposed a DWT-based image compression system implemented completely by deep convolutional neural networks. Haar wavelet is used to decompose the input image into sub-bands, then the sub-bands are encoded and decoded respectively in the wavelet transform domain and finally yield the reconstructed image by IDWT. In addition, a prediction model for high frequency sub-bands was used both in the encoder and the decoder, which can effectively remove the redundancy between low frequency sub-band and high frequency sub-bands. Experimental results show that the proposed compression approach outperforms JPEG, JPEG2000 and BPG, as well as some deep image compression approaches trained in spatial pixel domain. The proposed method shows advantage in the HF reconstruction, because the channel allocation between sub-bands can be more flexibly adjusted, making it possible to allocate more channels for HF sub-bands. As a result, it produces the reconstructed image with finer details and clearer textures, showing significant advantages in visual comparison.

Future works may consider other kinds of wavelets, such as Daubechies(dbN), which is adopted in the JPEG2000, or Symlet(symN). Furthermore, the correlation between different high frequency sub-bands can be used to improve the prediction model to eliminate more redundancy. Also, the correlation between sub-bands can be considered in the probability model to further increase the compression efficiency.

REFERENCES

[1] R. Li, W. Lu, H. Liang, Y. Mao, and X. Wang, "Multiple features with extreme learning machines for clothing image recognition," *IEEE Access*, vol. 6, pp. 36283–36294, 2018.
[2] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 6, pp. 1452–1464, Jun. 2018.
[3] B. S. Riggan, C. Reale, and N. M. Nasrabadi, "Coupled auto-associative neural networks for heterogeneous face recognition," *IEEE Access*, vol. 3, pp. 1620–1632, 2015.
[4] J. Yang, S. Li, and W. Xu, "Active learning for visual image classification method based on transfer learning," *IEEE Access*, vol. 6, pp. 187–198, 2017.
[5] M. Zhang, W. Li, and Q. Du, "Diverse region-based CNN for hyperspectral image classification," *IEEE Trans. Image Process.*, vol. 27, no. 6, pp. 2623–2634, Jun. 2018.
[6] Y. Li and L. Shen, "cC-GAN: A robust transfer-learning framework for HEp-2 specimen image segmentation," *IEEE Access*, vol. 6, pp. 14048–14058, 2018.
[7] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2018.
[8] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning Internal Representations by Error Propagation*. Cambridge, MA, USA: MIT Press1985.
[9] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
[10] R. Salakhutdinov and G. Hinton, "An efficient learning procedure for deep Boltzmann machines," *Neural Comput.*, vol. 24, no. 8, pp. 1967–2006, Aug. 2012.
[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE.*, vol. 11, no. 86, pp. 2278–2324, Nov. 1998.
[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 2012, pp. 1097–1105.
[13] G. Toderici *et al.* (2016). "Variable rate image compression with recurrent neural networks." [Online]. Available: https://arxiv.org/abs/1511.06085
[14] A. Graves. (2013). "Generating sequences with recurrent neural networks." [Online]. Available: http://arxiv.org/abs/1308.0850
[15] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra, "DRAW: A recurrent neural network for image generation," in *Proc. ICML*, Jul. 2015, pp. 1462–1471.
[16] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Trans. Consum. Electron.*, vol. 38, no. 1, pp. 18–34, Feb. 1992.
[17] G. Toderici *et al.*, "Full resolution image compression with recurrent neural networks," in *Proc. CVPR*, Jul. 2017, pp. 5435–5443.
[18] L. Theis, W. Shi, A. Cunningham, and F. Huszér, "Lossy image compression with compressive autoencoders," Palais des Congrés Neptune, Toulon, France, Tech. Rep. ICLR2017, 2017.
[19] M. Covell *et al.* (2017). "Target-quality image compression with recurrent, convolutional neural networks." [Online]. Available: https://arxiv.org/abs/1705.06687
[20] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimization of nonlinear transform codes for perceptual quality," in *Proc. PCS*, Nuremberg, Germany, Dec. 2016, pp. 1–5.
[21] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end Optimized Image Compression," in *Proc. ICLR*, Aug. 2017, pp. 1–27.
[22] F. Jiang, W. Tao, S. Liu, J. Ren, X. Guo, and D. Zhao, "An end-to-end compression framework based on convolutional neural networks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 10, pp. 3007–3018, Oct. 2018.
[23] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. V. Gool, "Conditional probability models for deep image compression," in *Proc. CVPR*, Feb. 2018, pp. 4394–4402.
[24] M. Li, W. Zuo, S. Gu, D. Zhao, and D. Zhang, "Learning convolutional networks for content-weighted image compression," in *Proc. CVPR*, Mar. 2018, pp. 3214–3223.
[25] K. Gregor, F. Besse, D. J. Rezende, I. Danihelka, and D. Wierstra, "Towards conceptual compression," Barselona, Spain, Tech. Rep. NIPS2016, 2016.
[26] A. S. Lewis and G. Knowles, "Image compression using the 2-D wavelet transform," *IEEE Trans. Image Process.*, vol. 1, no. 2, pp. 244–250, Apr. 1992.
[27] M. A. Losada, G. Tohumoglu, D. Fraile, and A. Artés, "Multi-iteration wavelet zero-tree coding for image compression," *Signal Process.*, vol. 80, no. 7, pp. 1281–1287, Jul. 2000.
[28] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The JPEG 2000 still image compression standard," *IEEE Signal Process. Mag.*, vol. 18, no. 5, pp. 36–58, Sep. 2001.
[29] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.
[30] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Trans. Comput.*, vol. C-23, no. 1, pp. 90–93, Jan. 1974.
[31] Y. Yang, N. P. Galatsanos, and A. K. Katsaggelos, "Regularized reconstruction to reduce blocking artifacts of block discrete cosine transform compressed images," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 3, no. 6, pp. 421–432, Dec. 1993.
[32] (2018). *BPG Format*. [Online]. Available: https://bellard.org/bpg
[33] R. S. Stanković and B. J. Falkowski, "The Haar wavelet transform: Its status and achievements," *Comput. Electr. Eng.*, vol. 29, no. 1, pp. 25–44, Jan. 2003.
[34] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. CVPR*, Miami, FL, USA, Jun. 2009, pp. 248–255.
[35] (2009). *Kodak Photocd Dataset*. [Online]. Available: http://r0k.us/graphics/kodak
[36] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," in *Proc. ACSSC*, Pacific Grove, CA, USA, Mar. 2003, pp. 1398–1402.

[37] D. P. Kingma and J. Ba. (2015). "Adam: A method for stochastic optimization." [Online]. Available: https://arxiv.org/abs/1412.6980.
[38] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. ICML*, Lille, France, Jun. 2015, pp. 448–456.
[39] *IrfanView*. Accessed: Feb. 16, 2019. [Online]. Available: http://www.irfanview.com
[40] (2016). *JPEG2000 for Matlab Ver. 1.22*. [Online]. Available: https://github.com/nsprljan/ImageCodingResearchTools
[41] O. Rippel and L. Bourdev, "Real-time adaptive image compression," in *Proc. ICML*, Long Beach, CA, USA, Jun. 2017, pp. 10–19.



CHUXI YANG was born in Changchun, China, in 1989. She received the B.S. degree in communication engineering, and the M.S. degree in communication and information systems from Jilin University, Changchun, China, in 2012 and 2015, respectively, where she is currently pursuing the Ph.D. degree in communication and information systems.

Her research interests include image coding, image compression based on neural networks, and wavelet transform-based image compression.

YAN ZHAO (M'10) was born in Jilin, China, in 1971. She received the B.S. degree in communication engineering from the Changchun Institute of Posts and Telecommunications, in 1993, the M.S. degree in communication and electronic from the Jilin University of Technology, in 1999, and the Ph.D. degree in communication and information systems from Jilin University, in 2003.

She was a Postdoctoral Researcher with the Digital Media Institute, Tampere University of Technology, Finland, in 2003. In 2008, she was a Visiting Professor with the Institute of Communications and Radio-Frequency Engineering, Vienna University of Technology. She is currently a Professor of communication engineering. Her research interests include image and video processing, multimedia signal processing, and error concealment for audio and video transmitted over unreliable networks.

SHIGANG WANG was born in Jilin, China, in 1962. He received the B.S. degree from Northeastern University, in 1983, the M.S. degree in communication and electronics from Jilin University of Technology, in 1998, and the Ph.D. degree in communication and information system from Jilin University, in 2001.

He is currently a Professor of communication engineering. His research interests include image and video coding, multidimensional signal processing, and stereoscopic and multi-view video coding.