End-to-End Optimized Versatile Image Compression With Wavelet-Like Transform

Haichuan Ma, Dong Liu, *Senior Member, IEEE*, Ning Yan, Houqiang Li, *Senior Member, IEEE*, and Feng Wu, *Fellow, IEEE*

Abstract—Built on deep networks, end-to-end optimized image compression has made impressive progress in the past few years. Previous studies usually adopt a compressive auto-encoder, where the encoder part first converts image into latent features, and then quantizes the features before encoding them into bits. Both the conversion and the quantization incur information loss, resulting in a difficulty to optimally achieve arbitrary compression ratio. We propose iWave++ as a new end-to-end optimized image compression scheme, in which iWave, a trained wavelet-like transform, converts images into coefficients without any information loss. Then the coefficients are optionally quantized and encoded into bits. Different from the previous schemes, iWave++ is *versatile*: a single model supports both lossless and lossy compression, and also achieves arbitrary compression ratio by simply adjusting the quantization scale. iWave++ also features a carefully designed entropy coding engine to encode the coefficients progressively, and a de-quantization module for lossy compression. Experimental results show that lossy iWave++ achieves state-of-the-art compression efficiency compared with deep network-based methods; on the Kodak dataset, lossy iWave++ leads to 17.34 percent bits saving over BPG; lossless iWave++ achieves comparable or better performance than FLIF. Our code and models are available at https://github.com/mahaichuan/Versatile-Image-Compression

Index Terms—Deep network, end-to-end optimization, image compression, lossless compression, lossy compression, wavelet transform

1 Introduction

With the development of deep learning, end-to-end optimized image compression has received much attention in recent years [1], [2], [3], [4], [5], [6], [7], [8], [9]. Owing to the powerful ability of representation learning, end-to-end compression methods [2], [6], [7] not only achieve comparable compression performance than BPG, which is the most advanced non-deep image compression method, but also demonstrate great potential to achieve lower time complexity when implemented with efficient parallel computing [1], [7]. Another appealing feature of end-to-end image compression is that it can be easily optimized for any differentiable loss function, while such optimization can be quite time consuming in non-deep methods [10].

The existing end-to-end image compression studies are inspired by the paradigm of the auto-encoder, which was first proposed in the well-known work of Hinton and Salakhutdinov [11]. In the original auto-encoder, the encoder part converts the input signal (e.g., an image) into latent features, while the decoder part is used to reconstruct the input. For the compression purpose, quantization is introduced into the auto-encoder to compress the latent features. The entire auto-encoder is trained to minimize the joint rate-distortion loss: $L = D + \lambda R$, where D stands for the distortion, such as mean-squared error, R stands for the bitrate of the quantized latent features, and λ is used to tradeoff the two [1], [2], [3], [4], [5], [6], [7], [8], [9].

The original auto-encoder [11] was proposed for dimensionality reduction. Accordingly, most of the existing studies about end-to-end image compression set the dimension of latent features to be less than the number of pixels, achieving *compact* representations. Such conversion from pixels to features incurs information loss, which implies that the existing work cannot deal with lossless image compression.

A more severe limitation of the compressive auto-encoder is that it has difficulty to achieve arbitrary compression ratio optimally. According to the joint rate-distortion loss, a specific compression ratio is corresponding to a specific value of λ. To achieve the optimal rate-distortion tradeoff, the entire auto-encoder is optimized end-to-end. Thus, different λ values lead to different trained models. As noted, the conversion from pixels to features is lossy, which implies that the lost information is different when λ changes. Indeed, it was verified that an end-to-end model trained with a specific λ performs badly for other compression ratios [12]. Consequently, it needs to store multiple models for different compression ratios, which lead to model storage overhead. It also brings difficulty in controlling the bitrate accurately.

When we look back at the traditional non-deep methods, we notice that they take a different approach. In JPEG-2000 [13], there is a *transform* step that converts image into coefficients. Different from the compressive auto-encoder, the number of coefficients is equal to the number of pixels, and the transform itself does not lose any information. It is possible to reconstruct the image perfectly from the coefficients.

• *The authors are with the CAS Key Laboratory of Technology in Geo-Spatial Information Processing and Application System, University of Science and Technology of China, Hefei 230027, China. E-mail: {hcma, nyan}@mail.ustc.edu.cn, {dongeliu, lihq, fengwu}@ustc.edu.cn.*

*Manuscript received 23 Dec. 2019; revised 7 Sept. 2020; accepted 19 Sept. 2020. Date of publication 23 Sept. 2020; date of current version 3 Feb. 2022.
(Corresponding author: Dong Liu.)
Recommended for acceptance by R. Timofte.
Digital Object Identifier no. 10.1109/TPAMI.2020.3026003*

TABLE 1
Differences Between the iWave in [16] and the iWave in This Paper

	iWave in [16]	iWave in this paper
Lifting structure	Update-first (Fig. 1b in [16])	Prediction-first (Fig. 2)
Optimization strategy	Non-end-to-end	End-to-end
Loss function for training	Partial reconstruction loss	Rate-distortion loss
Scaling after transform	Per-subband scaling	Uniform scaling
Lossless/lossy coding	Only lossy	Both lossless and lossy

Thus, JPEG-2000 supports lossless compression. For lossy compression, JPEG-2000 performs quantization on the coefficients, and achieves different compression ratios by adjusting the quantization scale. Similarly in H.264/AVC [14] and H.265/HEVC [15], the prediction and transform steps are perfectly revertible, it is only the quantization of coefficients that incurs information loss and achieves different compression ratios.

Using a revertible transform in end-to-end image compression has several important advantages. Essentially, an image is converted into coefficients without information loss, so the compression scheme natively supports lossless compression. It meanwhile supports lossy compression by simply quantizing the coefficients. It is only the quantization that incurs distortion. It is the quantized coefficients that determines the rate. Therefore, joint rate-distortion optimization is actually performed in the coefficient domain. Concretely, a specific λ value should correspond to a specific quantization scale Q_{step}. Ideally, we can have only one model but multiple quantization scales to achieve different compression ratios. Adjusting the quantization scale is convenient to control the bitrate. In short, we have a *versatile* compression scheme.

Can we have a revertible "transform" for end-to-end image compression? If we can, how will it perform? In our previous work [16], we had proposed a revertible transform termed iWave. Specifically, traditional wavelet transform can be implemented with the lifting scheme, and the lifting scheme itself is information lossless [17], [18]. Our idea was to replace the filters in the lifting scheme with trained convolutional neural networks (CNNs), resulting in wavelet-like transform–iWave. When we used well trained iWave to replace the traditional wavelets in JPEG-2000, we achieve better compression performance. However, iWave-enhanced JPEG-2000 does not catch up with the state-of-the-art methods, like BPG and newest end-to-end methods. And we had not explored lossless compression in [16].

In this paper, we integrate iWave into an end-to-end optimized scheme to further explore its potential. We find that quantization and entropy coding are two important modules that contribute significantly to the compression performance. First, since iWave is a wavelet-like transform, there is remaining correlation between the coefficients within each subband and among multiple subbands. Such correlation can be utilized in the entropy coding step. Thus, we devise a dedicated entropy coding module based on recurrent network and autoregressive model. Second, although vector quantization is better than scalar quantization in theory, the former is more difficult to train. We choose to scalar quantize the coefficients but to introduce a de-quantization module after scalar inverse-quantization. We observe that this module effectively compensates the quantization error. With the two modules and iWave all trained in a holistic network, we obtain iWave++ as a new end-to-end optimized image compression scheme.

We want iWave++ to be a versatile image compression solution, so we experiment in both lossy and lossless cases. We consider different configurations where we train a single model for both lossless and lossy, or separate models. Our experimental results show that lossy iWave++ achieves state-of-the-art compression efficiency compared with the other end-to-end methods; lossy iWave++ outperforms BPG significantly; lossless iWave++ achieves comparable or higher compression ratio than its non-deep rivals.

It is worth noting that our previous work in [16] and the work reported in this paper both adopt trained wavelet-like transform (iWave), but have different intentions. In [16], we intended to replace the linear wavelet transform in JPEG-2000 with iWave, so the problem of interest was how to adjust iWave to cooperate well with the other coding tools in JPEG-2000. In this paper, we intend to build an end-to-end optimized versatile compression scheme based on iWave, thus we use a simple as possible design for iWave but focus on how to devise efficient quantization, de-quantization, and entropy coding tools to realize the full potential of iWave. Accordingly, the iWave in [16] and the iWave in this paper have several differences as listed in Table 1.

The remainder of this paper is organized as follows. Section 2 reviews related work. In Section 3, we present the overview of iWave++, as well as the details of every module. Section 4 discusses the configurations and training of iWave++. In Section 5, we show the compression performance of iWave++ in comparison to the previous work. Section 6 concludes this paper.

2 Related Work

In this section, we introduce related work at two aspects. The first is end-to-end image compression, which is an emerging topic of great interest in recent years. The second is lifting scheme of wavelet transform, which inspires our work.

2.1 End-to-End Image Compression

End-to-end image compression has been rapidly developed in the past few years. To the best of our knowledge, before iWave++, there is no existing work of end-to-end image compression that handles both lossy and lossless cases. Thus, we review the two cases separately in the following.

2.1.1 *Lossy Image Compression*

Most of the existing studies focus on lossy image compression with the idea of compressive auto-encoder. The encoder part converts images to latent features. The latent features are further quantized and entropy encoded to obtain bitstream. According to the used neural network structure, these methods can be divided into two categories: recurrent neural network (RNN) based [3], [9], [19] and CNN based [1], [2], [6], [8].

Toderici *et al.* [19] proposed the first RNN-based end-to-end image compression method, which is able to provide scalable bitstream relying on iteratively compressing image or residual. Later on, two enhanced versions [3], [9] improved its performance with more efficient entropy model, SSIM-weighted loss, hidden-state priming, and bit allocation. A common feature of these studies is to convert images into binary latent features, which simplifies the entropy coding process.

Ballé *et al.* [1] proposed a general fully CNN-based end-to-end image compression framework. They introduced generalized divisive normalization [20] into the end-to-end model to remove the redundancy between channels. They used integer quantization instead of direct binarization. They also used piecewise linear functions to estimate the bitrate during training. By introducing hyper-prior [2] and autoregressive prior [6], both of which compress the latent features more efficiently, the CNN-based method in [6] is reported to outperform BPG when evaluated with RGB-PSNR, for the first time. More recently, there are several studies proposing to augment attention mechanism [21], [22] and multi-scale connections [23], [24] into this end-to-end framework.

Some improved methods have been proposed for the CNN-based end-to-end scheme. Li *et al.* [4] introduced the importance map to achieve automatic bit allocation. The importance map is also used to estimate the bitrate during training. Mentzer *et al.* [5] also used the important map, but avoided the transmission of the side information of the map. Theis *et al.* [8] proposed to use rounding operation for quantization, and used Gaussian scale mixtures to estimate the bitrate. Ripple *et al.* [7] embedded the pyramid decomposition into the end-to-end scheme. Agustsson *et al.* [25] proposed to use soft-to-hard vector quantization on the latent features, which improved the compression performance and facilitated the backpropagation in training.

It is worth noting that several studies try to achieve variable rate image compression. The RNN-based methods [3], [9], [19] naturally provide this functionality. Besides, Cai *et al.* [26] proposed to use multi-scale decomposition. They also provided bit allocation algorithms to determine the optimal scale of each image block for either a target rate or a target quality. Dumas *et al.* [27] investigated the possibility of using different quantization steps for the latent features to achieve different bitrates. Choi *et al.* [12] used a conditional auto-encoder where λ is regarded as the prior condition to achieve variable rate compression.

Compared with the aforementioned work, our proposed iWave++ is unique since the proposed iWave transform does not lose any information. In addition, iWave as a wavelet-like transform natively supports multi-scale pyramid decomposition and scalable bitstream.

2.1.2 Lossless Image Compression

Lossless coding is also known as entropy coding, which relies on accurate probability estimation of the information source. Thus, likelihood-based generative models [28], [29], [30] can be directly used for lossless image compression, as long as using an arithmetic encoder/decoder.

Mentzer *et al.* [31] proposed a fast and efficient lossless image compression scheme via a hierarchical model, which has shown comparable or better performance than traditional lossless image compression methods, such as PNG, WebP, JPEG-2000, and FLIF.

More recently, there are several studies using the flow-based models for lossless image compression [32], [33]. Actually, the flow is quite similar to the lifting scheme of wavelet transform, which we will detail later. Conceptually, our iWave ++ in the lossless case is very similar to the flow-based models. Nonetheless, we will show that iWave++ is also capable in lossy compression, which was not reported in [32], [33].

2.2 Lifting Scheme of Wavelet Transform

The lifting scheme of wavelet transform is also known as the second generation of wavelets, which is a custom-design construction of wavelets [17], [18]. Lifting scheme not only facilitates implementation, but also provides a high degree of freedom for designing new wavelets. Indeed, prior to the lifting scheme, handcrafting wavelets that satisfy the proper characteristics is not an easy task. Lifting scheme suggests a different way of cascading simple filters to realize sophisticated wavelets.

The lifting scheme[1] is composed by three steps: split, prediction, and update. Taking one-dimensional signal x as example, the signal is split into two parts, the even part, x_e, and the odd part, x_o. There is strong correlation between x_e and x_o, which can be used to predict x_o using x_e (or vice versa)

$$h = x_o - P(x_e), \tag{1}$$

where $P()$ denotes a specific filter used for prediction, and h is the prediction residual. Usually, h contains the high-frequency information of x. Next, an update step is performed

$$l = x_e + U(h), \tag{2}$$

where $U()$ stands for another specific filter used for update, and l is believed to contain low-frequency information of x. Prediction and update constitute a basic lifting step, and multiple lifting steps can be employed in the lifting scheme. Indeed, the above process provides a transform from x to l and h. Given properly chosen prediction and update filters, the transform is a specific kind of wavelets [18].

The lifting scheme is lossless in theory. Once we have l and h, immediately we can reconstruct x

$$x_e = l - U(h) \tag{3}$$

$$x_o = h + P(x_e) \tag{4}$$

$$x = M(x_e, x_o), \tag{5}$$

1. We use the prediction-first lifting structure in this paper. There is another kind of lifting structure known as update-first, for which we refer the reader to [16].

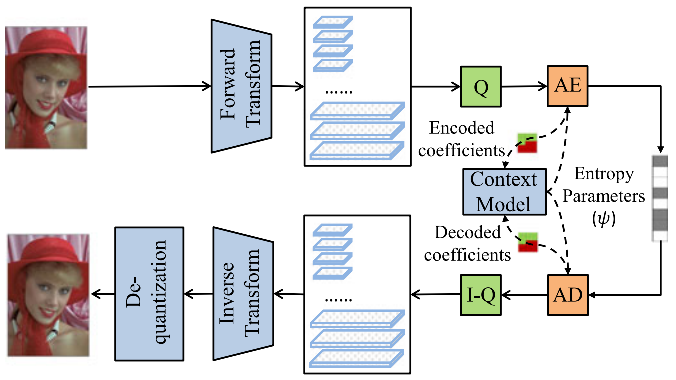


Fig. 1. Overview of the proposed iWave++. “Q” and “I-Q” stand for quantization and inverse-quantization, “AE” and “AD” stand for arithmetic encoding and arithmetic decoding, respectively. The forward transform and inverse transform modules are implemented with the lifting scheme and share the same set of parameters. The neural network-based context model produces entropy parameters for encoding/decoding the coefficients based on previously encoded/decoded coefficients. iWave++ supports lossless compression by discarding the quantization, inverse-quantization, and de-quantization modules.

where $M()$ stands for merging operation, i.e., the reverse of split. The above process is indeed an inverse wavelet transform once the prediction and update filters are properly chosen.

With the lifting scheme, several researchers have proposed to train the prediction and update filters for specific content, such as images [34]. In our previous work [16], we explored the idea of using trained CNNs as the prediction filters. The CNN-enhanced lifting scheme is termed iWave. In this paper, we inherit the idea of iWave, and add entropy coding and de-quantization modules to make an end-to-end optimized scheme, which is termed iWave++.

Inspired by the lifting scheme, several recent studies have focused on revertible networks that do not lose information during inference, such as i-RevNet [35]. The motivation is to decrease memory cost during network training and to reserve the input information as much as possible. In this paper, the proposed iWave is also a revertible network, but our focus is an end-to-end image compression scheme.

3 Proposed Method

In this section, we present iWave++ as a versatile end-to-end image compression framework. First, we introduce the overall structure of iWave++ for both lossy and lossless image compression. Second, the forward transform and inverse transform modules of iWave++ are introduced. Third, we introduce the carefully designed entropy model to efficiently compress the coefficients produced by iWave. Fourth, the de-quantization module that is used to compensate the quantization error is introduced.

3.1 Overview of iWave++

The proposed framework, iWave++, is illustrated in Fig. 1. There are four neural network-based modules: forward transform, inverse transform, de-quantization, and context model for entropy coding. An input image, x, is first converted to coefficients y via the forward transform module g_a: $y = g_a(x)$. Different from the previous end-to-end methods, the forward transform incurs no information loss, which is important for iWave++ to support both lossless and variable rate compression. Specifically, when we ensure that the forward transform operates in the integer domain, we can realize lossless compression. When we adjust the quantization of the coefficients, while keeping the other modules unchanged, we can achieve variable rate compression. Since the rate is solely dependent on the quantized coefficients, we can achieve virtually any reachable bitrate by simply adjusting the quantization step.

For lossy compression, the coefficients y are scalar quantized to obtain the discrete indexes q

$$q = Q(y) = [y/Q_{step}], \tag{6}$$

where Q_{step} stands for the quantization step, the unique parameter to control rate and distortion in our scheme. For lossless compression, $q = y$.

Next, the (quantized) coefficients q will be compressed into bits by an arithmetic encoder with the help of the neural network-based context model. Conventional wavelet coefficients coding methods, such as EZW [36], SPIHT [37], and EBCOT [38], are all handcrafted with the intuition that correlation exists within and across subbands. Similarly, we also want to exploit the correlation, but we choose to build an efficient context model for arithmetic coding based on trained neural networks.

At the decoder side, an arithmetic decoder is used to decompress the bitstream, followed by a simple inverse-quantization to obtain the reconstructed coefficients $\hat{y}$

$$\hat{y} = q \times Q_{step}, \tag{7}$$

or $\hat{y} = q = y$ for lossless compression. $\hat{y}$ is then converted to the reconstruction image $\hat{x}$ by using the inverse transform g_s: $\hat{x} = g_s(\hat{y})$. Note that g_s is the exact inverse of g_a, i.e., the inverse transform module shares the same set of parameters with the forward transform module. Therefore, the information loss due to quantization, incurred after the forward transform, cannot be compensated by the inverse transform. Then for lossy compression, a de-quantization module is introduced to compensate the quantization loss.

3.2 iWave: Wavelet-Like Transform

In previous end-to-end image compression methods, the auto-encoder is usually implemented by stacking multiple convolutional layers [1], [2], [6]. These encoders are not revertible. On the contrary, our iWave is based on the lifting scheme that is lossless, as mentioned in Section 2.2.

We use one-dimensional signal as example. As shown in Fig. 2a, after one-step lifting (i.e., a prediction and an update), we obtain l_1 and h_1. There are N lifting steps, each of which may use a different prediction/update filter, as indicated by P_i/U_i. In iWave, P_i and U_i are all implemented by CNNs, whose parameters will be optimized through the end-to-end training. In this paper, we adopt a uniform structure for P_i and U_i, which is depicted in Fig. 3. Note that $tanh$ is used as the activation function, which leads to better results than ReLU in our experiments. After N lifting steps, we obtain l_N and h_N as the low-frequency component and the high-frequency component of wavelet coefficients, respectively.

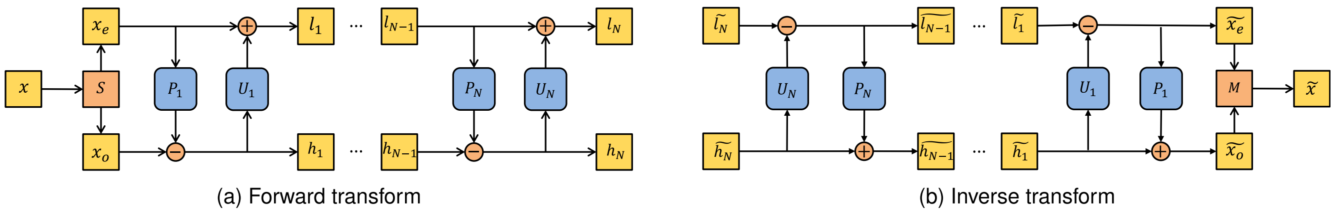


Fig. 2. iWave forward transform and inverse transform implemented with the lifting scheme. S stands for split, M stands for merge, P_i and U_i stand for the ith prediction unit and the ith update unit, respectively, N is the total number of lifting steps. All the prediction units and update units are built on convolutional neural networks with trainable parameters. Note that for two-dimensional image, transform is performed twice, first row-wise and then column-wise, see Fig. 4 for details.

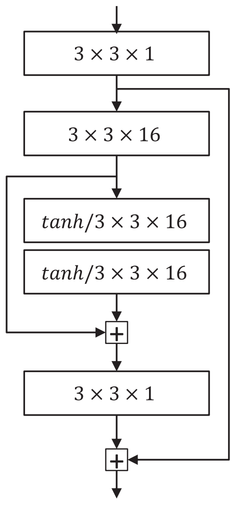


Fig. 3. The network structure used in this paper for each P_i or U_i (c.f. Fig. 2). The numbers like $3 \times 3 \times 16$ indicate the kernel size (3×3) and the number of channels (16) of each layer. $tanh$ indicates the adopted nonlinear activation function.

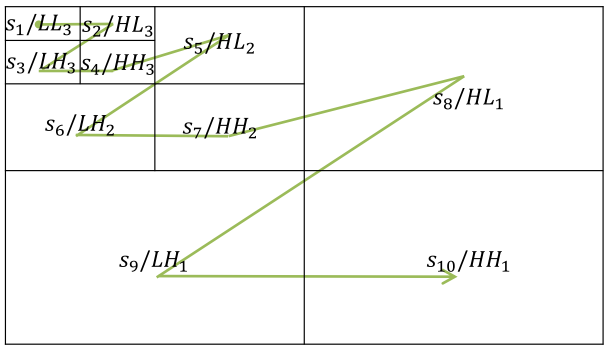


Fig. 5. The subbands obtained by $D = 3$ times of forward transform are shown for example. The encoding/decoding order of the subbands is denoted by the green line as well as by the subscripts of $s_1, s_2, \ldots, s_{3D+1}$.

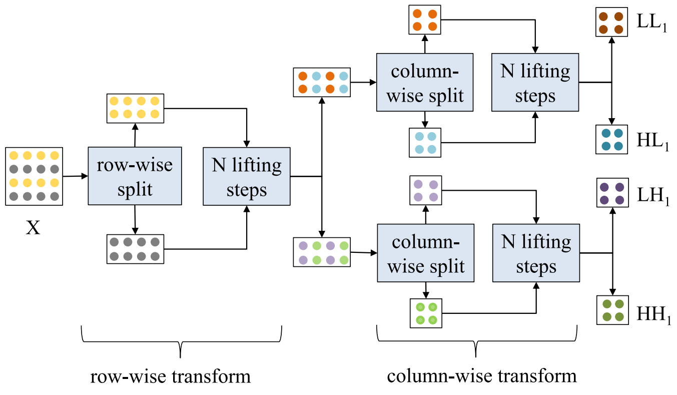


Fig. 4. Two-dimensional iWave forward transform implemented with the lifting scheme. Note that LL_1 may be further transformed to LL_2, HL_2, LH_2, HH_2, and LL_2 may be further transformed ..., which forms a pyramid. Stitching the subbands together, we have Fig. 5.

The inverse transform of iWave++ is strictly the reverse of the forward transform. As shown in Fig. 2b, it is realized by reversing the order of the operations and swapping the additions and subtractions. In iWave, we ensure the symmetry between forward transform and inverse transform. P_i and U_i in the inverse transform are also implemented by CNNs, whose parameters are exactly the same with those in the forward transform.

For two-dimensional image, the transform is performed twice, first row-wise and then column-wise, as depicted in Fig. 4. Here, $X, LL_1, HL_1, LH_1, HH_1$ are all two-dimensional; L stands for low-frequency and H stands for high-frequency. This is indeed a transform from X to four *subbands*: $\{LL_1, HL_1, LH_1, HH_1\}$.

Moreover, for natural images we often perform multi-scale pyramid decomposition, i.e., we forward transform the LL_1 subband to obtain $\{LL_2, HL_2, LH_2, HH_2\}$, and we forward transform the LL_2 subband, and so on. After D times of forward transform, we have $3D + 1$ subbands: $\{LL_D, HL_D, LH_D, HH_D, \ldots, HL_1, LH_1, HH_1\}$. In Fig. 5, we show the coefficient structure after three times of forward transform at different levels.

For lossless image compression, we need to work in the integer domain. However, trained CNNs often have floating-point numbers. Thus, we add a rounding operation after each P_i and U_i. For example, (1) and (2) can be changed to

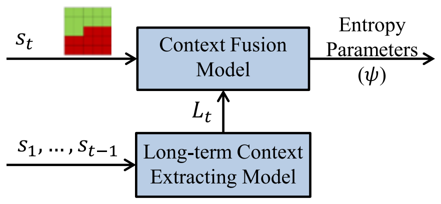
$$h = x_o - [P(x_e)] \tag{8}$$

$$l = x_e + [U(h)]. \tag{9}$$

Similar rounding operations are also employed in the inverse transform, thus ensuring perfect reconstruction. The rounding operations in (8) and (9) are performed in both training and inference for lossless iWave++. During training, gradient is back-propagated with no change through the rounding.

3.3 Entropy Coding

After the iWave forward transform, we optionally quantize the coefficients y to obtain the discrete indexes q, then we design an entropy coding module to deal with q. Note that the wavelet-like transform performs multi-scale pyramid



s_t
Context Fusion Model
Entropy Parameters (ψ)
L_t
$s_1, \ldots, s_{t-1}$
Long-term Context Extracting Model

Fig. 6. Network structure of the context model. The long-term context extracting model is implemented with a recurrent network of three LSTM layers, which generates the long-term context L_t based on all of the encoded/decoded subbands before (and including) s_{t-1}. The context fusion model is implemented with a CNN (e.g., Figs. 7 and 8), which fuses the information of the encoded/decoded coefficients (green area) of the current subband s_t with L_t to generate the final entropy parameters.

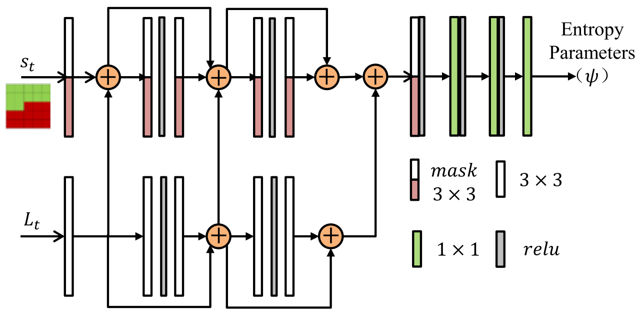


Fig. 7. The “heavy” version of the context fusion model used in the experiments by default. Note that mask convolution is used to process s_t since it has partial area unavailable (red area denotes unavailable).

decomposition. After D times of decomposition, there are $3D+1$ subbands. In this subsection, the subbands are denoted by $\mathcal{S} = \{s_1, s_2, \ldots, s_{3D+1}\}$, in the order of LL to HH. This order is defined according to the importance of the subbands in the wavelet transform theory. For example, the case of $D=3$ is depicted in Fig. 5. These subbands are compressed one by one in the specific order shown in Fig. 5. Within each subband, the coefficients are compressed in the scan-line order.

The entropy coding is fulfilled by a context model, which outputs a set of entropy parameters ψ, a probability model, which predicts the probability distribution of each coefficient $p(\cdot|\psi)$, and an arithmetic coding engine. Given the accurate probability distribution and a long enough stream, arithmetic coding is theoretically optimal. Note that we do not predict the probability distribution directly since its support is very large (the dynamic range of a coefficient can be larger than 6,000). Instead, we first predict a set of parameters and then parameterize the probability distribution. More accurately, we parameterize the cumulative distribution and then differentiate it (see below for details). This idea is also used in the recent literature [2].

Context Model. Essentially, the context model is a fully autoregressive model, which is depicted in Fig. 6. The long-term context extracting model is used to process the already coded subbands and to derive the long-term context L_t, and the context fusion model combines the information in L_t with the information of the already coded coefficients in the current subband.

For the long-term context extracting model, we use an RNN. Precisely, we use three convolutional long short-term memory (LSTM) layers [39], [40], [41]. The RNN helps use long-term memory without duplicate computation. When the entire subband s_{t-1} is finished, it is sent to the RNN to obtain L_t, and L_t remains unchanged when coding the next subband s_t. When the entire subband s_t is finished, it is sent to the RNN to update the output from L_t to L_{t+1}, which then serves for the next subband s_{t+1}. A minor tweak worth noting is that, when the resolutions of s_{t-1} and s_t are different, e.g., when coding s_5 in Fig. 5, L_t is up-sampled by a deconvolutional layer whose parameters are trainable. The states in the RNN, i.e., the internal variables in the three LSTM layers, are also up-sampled by deconvolutional layers.

For the context fusion model, we use a CNN that is depicted in Fig. 7, where we have used the mask convolution, also adopted in [5], to process s_t to ensure that the network only uses the already coded coefficients. We also try a light version for the CNN, which is depicted in Fig. 8.

Complexity of Context Model. According to the context model, all the coefficients must be processed sequentially in the specific order, since one coefficient relies on all the already coded coefficients. This can be quite time consuming. In practice, we may simplify the context model so that each coefficient relies on only partial coefficients. We will come back to this issue in Section 5.4.

Probability Model. The output of the context model is used to decide a probability distribution to be used in arithmetic coding. For the probability model, we use the formulation proposed in [2]. In detail, we parameterize a cumulative distribution and then differentiate it to obtain the required probability distribution. The cumulative distribution is

$$c(x) = f_K(f_{K-1}(\cdots(f_1(x)))), \tag{10}$$

where $K=5$ in this paper. The cascaded functions $f_k(x)$ are defined as follows. When $k=1$,

$$f_1(x) = g_1(\mathbf{H}^1 x + \mathbf{b}^1) \tag{11}$$

$$g_1(x) = x + \mathbf{a}^1 \cdot tanh(x), \tag{12}$$

where $\mathbf{H}^1, \mathbf{b}^1, \mathbf{a}^1 \in \mathbb{R}^{3\times1}$, the central dot stands for element-wise multiplication, and $tanh$ is applied point-wise. When $1 < k < K$,

$$f_k(x) = g_k(\mathbf{H}^k x + \mathbf{b}^k) \tag{13}$$

$$g_k(x) = x + \mathbf{a}^k \cdot tanh(x), \tag{14}$$

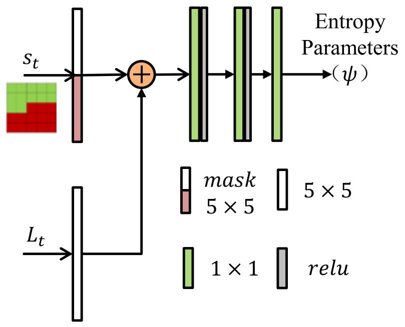


Fig. 8. The “light” version of the context fusion model. Please compare to Fig. 7.

where $\mathbf{H}^k \in \mathbb{R}^{3\times 3}$, $\mathbf{b}^k, \mathbf{a}^k \in \mathbb{R}^{3\times 1}$. When $k = K$,

$$f_K(x) = \text{sigmoid}(\mathbf{H}^K x + b^K), \tag{15}$$

where $\mathbf{H}^K \in \mathbb{R}^{1\times 3}$, $b^K \in \mathbb{R}$.

All of the $\mathbf{H}^k, \mathbf{b}^k/b^K, \mathbf{a}^k$ constitute the entropy parameters ψ. In this paper, these entropy parameters are obtained by the context model, i.e., they are different channels of the output of the context fusion model shown in Fig. 6. This is different from the design in [2], where the parameters are trained out and fixed during the inference. In other words, we use context-adaptive parameters for the probability model, whereas [2] did not enable context adaptivity.

Comparison to BinaryRNN [9]. In our scheme, we compress multiple subbands of coefficients, which are obtained by the wavelet-like transform and form a pyramid. In [9], BinaryRNN was proposed to compress multiple “layers” of binary codes, where the codes were obtained by another RNN that iterates multiple times. Our entropy coding tool and BinaryRNN have some properties in common, notably using RNN to compress the subbands/layers sequentially. As such, both our framework and [9] provide *scalable* coding: our framework provides spatial resolution scalability, whereas [9] provides signal-to-noise ratio scalability.

As for differences, first, note that the resolutions of our subbands are unequal, so we need to up-sample the output of the long-term context extracting model, as well as the states of the RNN, when necessary. This is also the most important reason why we have used RNN instead of concatenating all subbands into a cube.[2] Second, the coefficients in our framework are quantized to multi-ary integers rather than binary in [9].

3.4 De-Quantization

De-quantization can compensate the quantization loss and improve the quality of reconstructed image. Inspired by the image restoration network proposed in [43], we use a plain CNN as the de-quantization module, which is depicted in Fig. 9. The network contains a global shortcut as well as multiple residual blocks, which was observed to accelerate its training.

The de-quantization appears similar to decoder post-processing. Using post-processing to enhance image compression is well known and has been extensively studied [44], [45]. Nonetheless, de-quantization in iWave++ is distinct from post-processing in other end-to-end image compression schemes. Since iWave++ adopts revertible transform, its inverse transform shall be consistent with its forward transform. The quantization error introduced *after* forward transform cannot be compensated by the inverse transform. Thus, when using iWave++ for lossy compression, we propose the de-quantization module *after* inverse transform as a solution. The situation is quite different in other end-to-end schemes, where the decoding network is not constrained to be reversion of the encoding network. The decoding network, being well trained, has the capability of reducing quantization error. Thus, adding a de-quantization module is less important. Please refer to Section 5.3.3 for the related results.

2. The reader may check [42] for a cube-based entropy coding tool.

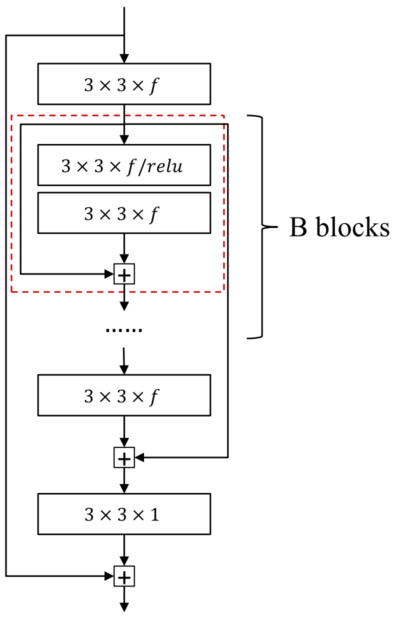


Fig. 9. The network structure used in this paper for the de-quantization module. The shown numbers like $3 \times 3 \times f$ indicate the kernel size (3×3) and the number of channels (f) of each layer. $relu$ indicates the used nonlinear activation function. There are B residual blocks.

We shall remark that de-quantization is not the only solution to the aforementioned problem of iWave++. Another possible solution is to untie the binding of forward transform and inverse transform. However, this is not a good solution if we want to support both lossless and lossy compression via using the same transform, or we want to support variable rate with the same transform.

4 Configurations and Training

In the JPEG-2000 standard, there are two kinds of wavelet transform, namely CDF 5/3 and CDF 9/7 [13]. CDF 5/3 wavelet has only integer arithmetics, while CDF 9/7 wavelet has floating-point numbers. Accordingly, for lossless compression CDF 5/3 is the only choice. But for lossy compression, CDF 9/7 is observed to outperform CDF 5/3. Inspired by this, we experiment several different configurations of iWave++, whose training targets are also different.

In the following, we use $g_a(\cdot)$ and $g_s(\cdot)$ to refer to the forward transform and inverse transform, which share the same set of parameters $\boldsymbol{\theta_t}$. $Q(\cdot)$ stands for quantization that relies on a single parameter Q_{step}. $R[\cdot]$ stands for estimating the bit-rate, which equals to

$$R[\mathcal{S};\boldsymbol{\theta_c}] = \sum_{s_i\in\mathcal{S}} \sum_{j=1}^{|s_i|} -\log{}_2 p(s_{ij};\boldsymbol{\theta_c}), \tag{16}$$

where s_{ij} is the jth coefficient in the subband s_i, and $\boldsymbol{\theta_c}$ includes all the parameters in the entropy coding module. $D(\cdot)$ stands for the de-quantization whose parameters are denoted by $\boldsymbol{\theta_d}$.

Lossless iWave++. The loss function to train the lossless iWave++ is

$$L(\boldsymbol{\theta_t},\boldsymbol{\theta_c}) = \sum_{x\in\mathcal{X}} R[g_a(x;\boldsymbol{\theta_t});\boldsymbol{\theta_c}], \tag{17}$$

where $\mathcal{X}$ is the training set.

Lossy Multi-Model iWave++. To train lossy iWave++, we may train multiple models each for a different bit-rate. This is also the common practice in most of the deep image compression methods [2], [6]. For each model we need to specify a Lagrange multiplier λ, which controls the tradeoff between rate and distortion. The loss function is

$$\begin{aligned} L(\boldsymbol{\theta_t},\boldsymbol{\theta_c},\boldsymbol{\theta_d},Q_{step}) = &\sum_{x\in\mathcal{X}} R[Q(g_a(x;\boldsymbol{\theta_t});Q_{step});\boldsymbol{\theta_c}] \\ &+\lambda d[x, D(g_s(Q(g_a(x;\boldsymbol{\theta_t});Q_{step});\boldsymbol{\theta_t});\boldsymbol{\theta_d})], \end{aligned} \tag{18}$$

where $d[\cdot,\cdot]$ stands for a difference function, e.g., mean-squared error (MSE).

Lossy Variable Rate iWave++. We also test the possibility of training single iWave++ model for variable rate lossy compression. For this configuration, our training strategy is virtually a multi-task learning one. We select several values as different Lagrange multipliers $\lambda_k, k=1,\ldots,K$, and the loss function is

$$\begin{aligned} L(\boldsymbol{\theta_t},\boldsymbol{\theta_c},\boldsymbol{\theta_d}) = &\sum_k\sum_{x\in\mathcal{X}} R[Q(g_a(x;\boldsymbol{\theta_t});Q^k_{step});\boldsymbol{\theta_c}] \\ &+\lambda_k d[x, D(g_s(Q(g_a(x;\boldsymbol{\theta_t});Q^k_{step});\boldsymbol{\theta_t});\boldsymbol{\theta_d})], \end{aligned} \tag{19}$$

where Q^k_{step} is the quantization scale corresponding to λ_k. Note that Q^k_{step} is used for training only. When using the trained model for lossy compression, one can set the quantization scale to arbitrary positive number. Moreover, one can set different quantization scales to different coefficients, although we use a uniform scalar quantization on all the coefficients in training.

Universal iWave++. Finally, we may train a single iWave++ model to support both lossy and lossless compression. This model is named universal iWave++. Note that this configuration is not possible in all of the previous end-to-end image compression methods. This configuration can also be regarded as a multi-task learning task. Accordingly, the loss function is

$$\begin{aligned} L(\boldsymbol{\theta_t},\boldsymbol{\theta_c},\boldsymbol{\theta_d}) = &\lambda_{\text{lossless}}\sum_{x\in\mathcal{X}} R[g_a(x;\boldsymbol{\theta_t});\boldsymbol{\theta_c}] \\ &+\sum_k\sum_{x\in\mathcal{X}} R[Q(g_a(x;\boldsymbol{\theta_t});Q^k_{step});\boldsymbol{\theta_c}] \\ &+\lambda_k d[x, D(g_s(Q(g_a(x;\boldsymbol{\theta_t});Q^k_{step});\boldsymbol{\theta_t});\boldsymbol{\theta_d})], \end{aligned} \tag{20}$$

where $\lambda_{\text{lossless}}$ is set to 1 in this paper.

5 Experimental Results

In this section, we first introduce the experimental settings. Then we present the compression performance of the four configurations of iWave++ that have been described in Section 4. We perform ablation studies to verify the effectiveness of the different modules in iWave++. Last but not the least, we analyze the computational complexity of iWave++, and suggest a possible speed-up scheme.

5.1 Settings

iWave++ supports both lossless and lossy compression, but all of the existing end-to-end methods support only one. To make fair comparisons with the previous work, we follow their training settings as many as possible. Thus, the training data for lossless and lossy compression are slightly different.

For training lossy (multi-model and variable rate) and universal iWave++, we use 800 high-resolution images in the DIV2K training set [46] and 585 high-resolution images in the CLIC challenge dataset.[3] MSE is used as the only distortion measure in the training.

For training lossless iWave++, we use three settings: DIV2K training set, ImageNet32 training set, and ImageNet64 training set. Each of the last two sets contains around 1.2 million images. The three settings are used for different test sets: DIV2K for the Kodak dataset, ImageNet32 and ImageNet64 for their corresponding test set, respectively.

All of the training images are converted to grayscale and then split into 128×128 blocks. The blocks are used as training data without any data augmentation.

During training, we divide the parameters into two sets: $\{\boldsymbol{\theta_t},\boldsymbol{\theta_d}\}$ and $\{\boldsymbol{\theta_c}\}$, and we optimize the two sets alternatively with two optimizers. Specifically, we keep the second set unchanged and train the first set for several mini-batches, and then keep the first set unchanged and train the second set for several mini-batches, and so on. This trick helps the training converge smoothly. The Adam algorithm [47] with its default settings is adopted for training. The learning rate is empirically set to 0.0001 constantly, as we find that reducing learning rate does not lead to performance improvement. We train the models for about 150 epochs. The training is implemented with TensorFlow.

In the testing stage, we convert RGB images into the YUV or YCoCg color space, and use the same model, which is trained on grayscale blocks, on the three channels individually. Decoded YUV/YCoCg images are converted back into the RGB color space. We have tried training different models for different channels, whose benefit was observed not significant. It is worth noting that iWave in this paper is designed for two-dimensional signal, so we need the color transform. In principle, iWave can also be designed for three-dimensional signal, such as RGB image, but that 3D iWave would be restricted to the specific color format. Therefore, 2D iWave plus a separate color transform is used by default in this paper.

5.2 Compression Performance

5.2.1 Lossy Multi-Model iWave++

First, we compare lossy multi-model iWave++ with several typical lossy image compression methods, including five non-deep methods: JPEG, WebP, JPEG-2000 (using the OpenJPEG software[4]), BPG, xvc,[5] and three deep methods denoted by *Variational* [2], *Joint* [6], and *iWave* [16]. Among them, xvc is the best non-deep method and *Joint* is among the best deep methods, to our knowledge. We use the Kodak image set (24 images of 768×512 resolution) and the Tecnick image set (100 images of 1200×1200 resolution) [48] as the test sets, both of which are commonly used as benchmark dataset. We report both PSNR and SSIM as quality metrics. For lossy iWave++, we convert the RGB images

3. http://www.compression.cc/challenge/
4. http://www.openjpeg.org/
5. https://xvc.io/

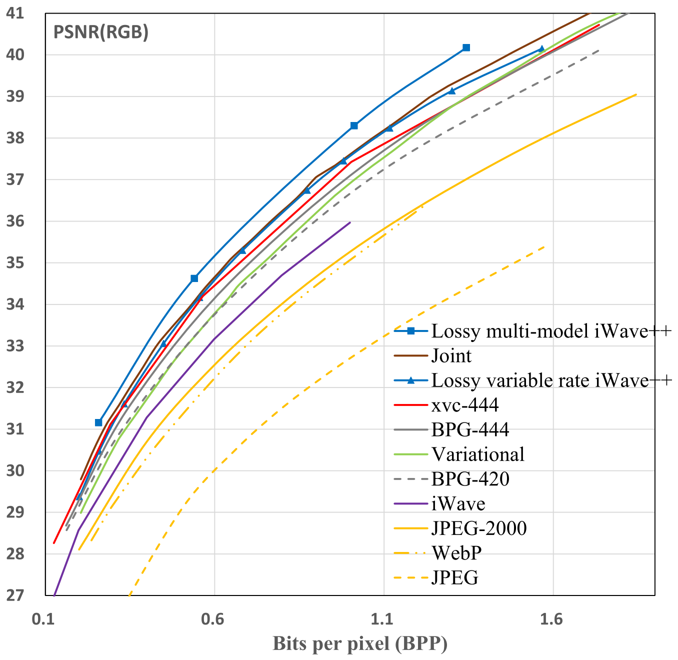


Fig. 10. Comparison of different lossy compression methods on the Kodak dataset. Shown are the average rate (bits-per-pixel) and the average PSNR (calculated in RGB) of the 24 images. "Joint" refers to [6], "Variational" refers to [2], and "iWave" refers to [16].

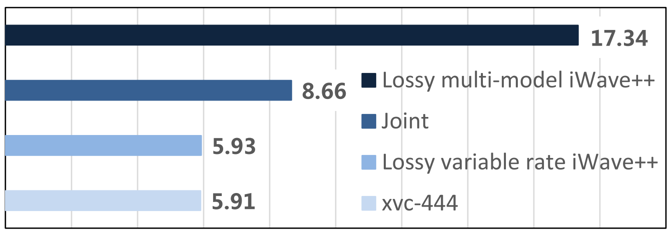


Fig. 12. Average BD-rate reduction ratios, using PSNR (RGB) as quality metric, of each method compared to BPG (4:4:4) on the Kodak dataset. "Joint" refers to [6].

into the YUV color space. Decoded YUV images are converted back into the RGB color space to calculate PSNR/SSIM.

The results measured by PSNR are shown in Figs. 10 and 11 for the two test sets respectively. Our lossy multi-model iWave++ preforms better than all of the other methods, including non-deep and deep methods, on both datasets. We use BPG (4:4:4) as the anchor, and calculate the average BD-rate reduction of another method over the anchor on the Kodak dataset. Fig. 12 shows that lossy multi-model iWave ++ achieves on average 17.34 percent BD-rate reduction. In comparison, *Joint* as a state-of-the-art deep method achieves 8.66 percent.

iWave++ significantly outperforms *iWave* as observed from Figs. 10 and 11. It indicates that the proposed entropy coding and de-quantization tools boost the compression efficiency with the help of end-to-end learning.

The results measured by SSIM are shown in Fig. 13 for the Kodak dataset. Note that the SSIM results of several compared methods are quoted from the supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2020.3026003, of [6]. Our lossy multi-model iWave++ is quite competitive. It performs on par with *Joint* (MS-SSIM optimized) and outperforms all the others. This result is somewhat surprising, since our iWave++ is optimized for MSE rather than for SSIM. In end-to-end methods, optimizing for MSE and for SSIM/MS-SSIM were believed quite different. For example one can compare the two curves in Fig. 13 marked with "Joint (MS-SSIM optimized)" and

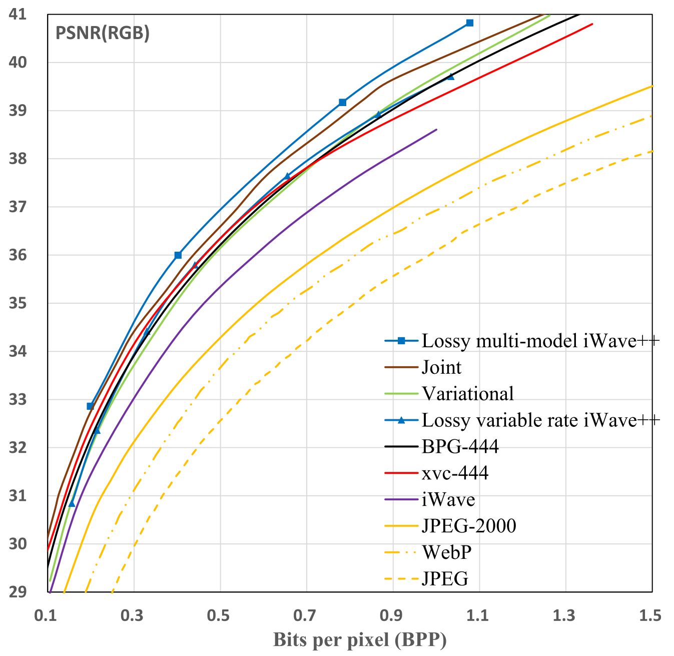


Fig. 11. Comparison of different lossy compression methods on the Tecnick dataset. Shown are the average rate (bits-per-pixel) and the average PSNR (calculated in RGB) of the 100 images. "Joint" refers to [6], "Variational" refers to [2], and "iWave" refers to [16].

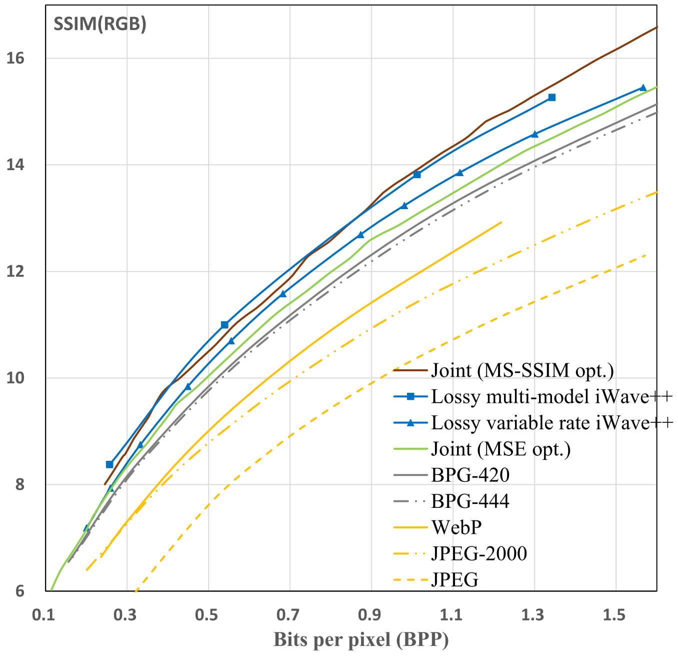
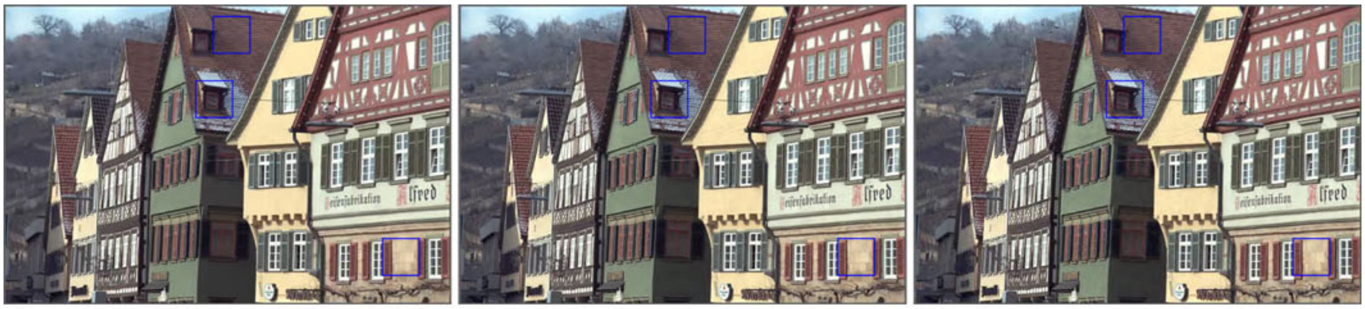
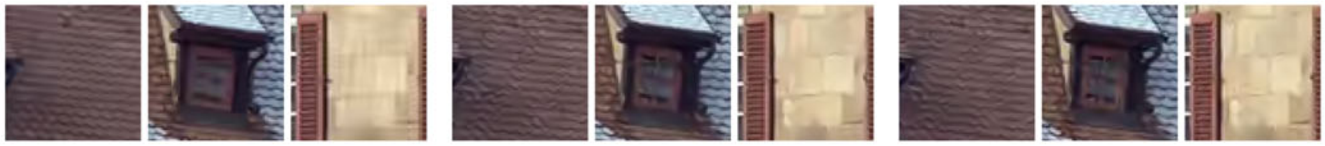
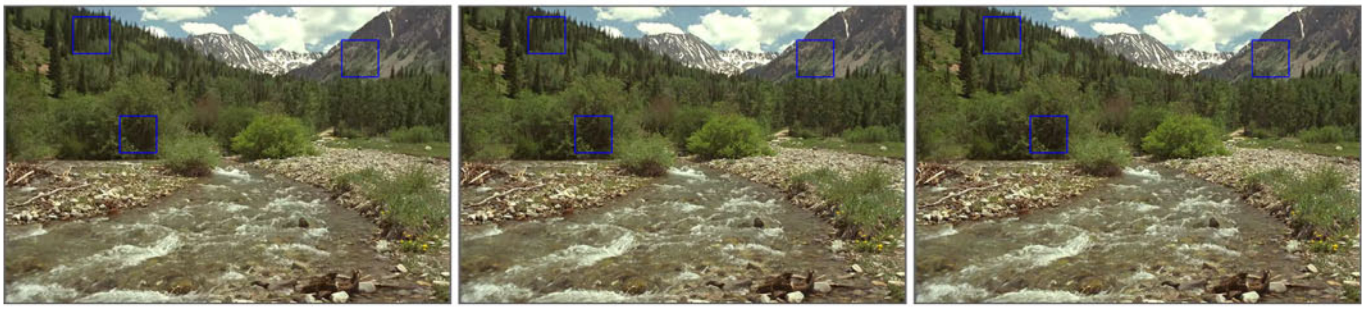
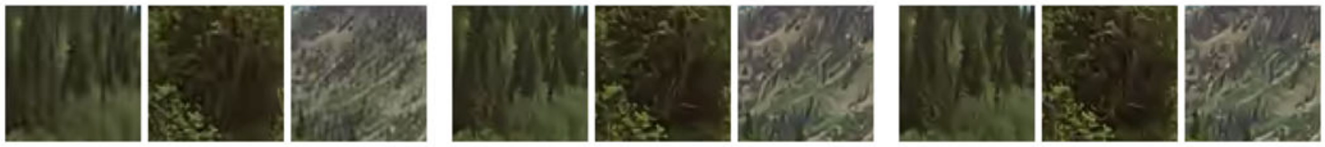


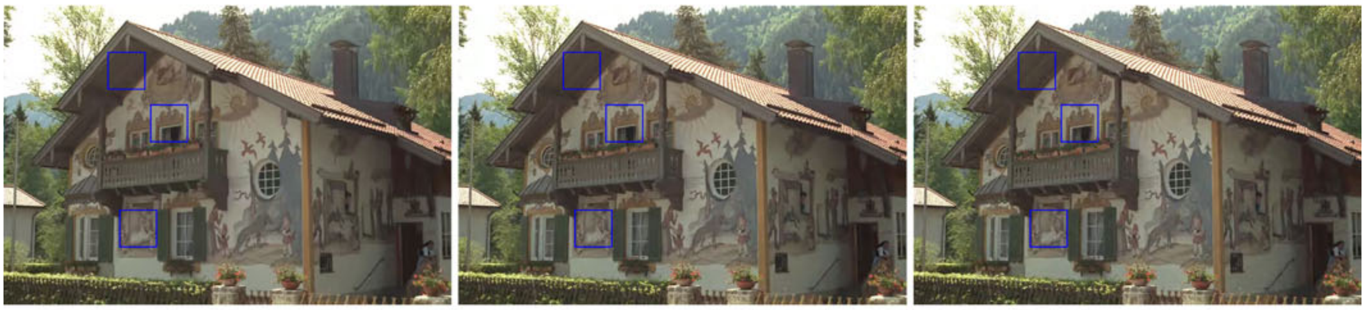
Fig. 13. Comparison of different lossy compression methods on the Kodak dataset. Shown are the average rate and the average SSIM (calculated in RGB and shown in dB: $SSIM_{dB} = 10\log_{10}(1 - SSIM)$) of the 24 images. "Joint" refers to [6] and "Variational" refers to [2].


(a) **Kodak08**. **Left: BPG**, Rate: 31677 bytes, PSNR: 29.59 dB, SSIM: 0.8814. **Middle: Lossy multi-model iWave++**, Rate: 31800 bytes, PSNR: 30.39 dB, SSIM: 0.9011. **Right: Lossy variable rate iWave++**, Rate: 31300 bytes, PSNR: 29.95 dB, SSIM: 0.8947


(b) **Insets of Kodak08**. **Left**: BPG, **Middle**: Lossy multi-model iWave++, **Right**: Lossy variable rate iWave++


(c) **Kodak13. Left: BPG**, Rate: 45126 bytes, PSNR: 28.28 dB, SSIM: 0.8259. **Middle: Lossy multi-model iWave++**, Rate: 45065 bytes, PSNR: 28.67 dB, SSIM: 0.8531. **Right: Lossy variable rate iWave++**, Rate: 45575 bytes, PSNR: 28.55 dB, SSIM: 0.8257


(d) **Insets of Kodak13**. **Left**: BPG, **Middle**: Lossy multi-model iWave++, **Right**: Lossy variable rate iWave++


(e) **Kodak24. Left: BPG**, Rate: 26150 bytes, PSNR: 30.15 dB, SSIM: 0.8581. **Middle: Lossy multi-model iWave++**, Rate: 26941 bytes, PSNR: 30.86 dB, SSIM: 0.8841. **Right: Lossy variable rate iWave++** Rate: 26070 bytes, PSNR: 30.43 dB, SSIM: 0.8732

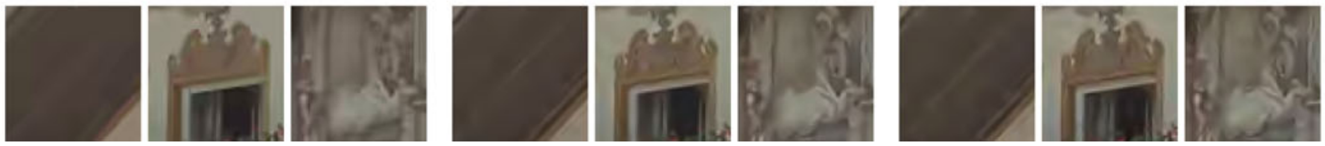

(f) **Insets of Kodak24. Left**: BPG, **Middle**: Lossy multi-model iWave++, **Right**: Lossy variable rate iWave++

Fig. 14. Visual results of BPG (4:4:4), lossy multi-model iWave++, and lossy variable rate iWave++.

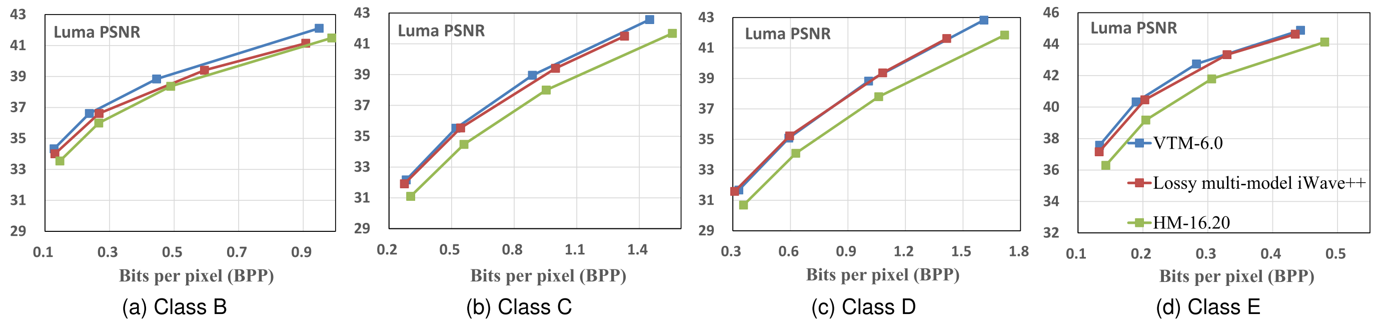


Fig. 15. Comparison between iWave++, H.265/HEVC (HM encoder), and H.266/VVC (VTM encoder). Shown are the average rate and the average luma PSNR of each class of the HEVC common test sequences.

TABLE 2
Average BD-Rate Reduction Ratios of Lossy Multi-Model iWave++ Compared to H.265/HEVC and H.266/VVC on the HEVC Common Test Sequences

	Over H.265/HEVC			Over H.266/VVC		
	Y	U	V	Y	U	V
Avg. Class B	– 12.50%	−25.66%	−27.85%	14.66%	−4.79%	−4.14%
Avg. Class C	−17.27%	−22.82%	−24.84%	6.17%	−5.96%	−3.64%
Avg. Class D	−19.12%	−24.87%	−25.74%	−0.51%	−12.32%	−9.86%
Avg. Class E	−21.75%	−29.78%	−27.55%	7.07%	−10.13%	−2.60%
Avg. All	**−17.17%**	**−25.51%**	**−26.32%**	**6.27%**	**−8.42%**	**−5.30%**

"Joint (MSE optimized)," and observe their performance gap. On the contrary, our lossy multi-model iWave++ is optimized only for MSE but still performs well in terms of SSIM.

Several reconstructed images are shown in Fig. 14 for visual comparison. Compared to BPG (4:4:4), the reconstructed images by our iWave++ have more details, less artifacts, and thus better subjective quality. This is in line with the PSNR and SSIM numbers.

Moreover, we compare lossy multi-model iWave++ with state-of-the-art video coding technology. In video coding schemes, the first frame of a video sequence is compressed like a still image. Therefore, we choose some video sequences and use different video coding schemes to compress their first frames. We also use lossy multi-model iWave++ to compress these frames for comparison. To ensure fair comparison, we use the common test sequences of H.265/HEVC and the upcoming H.266/VVC,[6] including Class B (3 sequences, 1920×1080), Class C (4 sequences, 832×480), Class D (4 sequences, 416×240), and Class E (3 sequences, 1280×720). We omit the other common test sequences because they have 10-bit color depth and our iWave++ currently supports 8-bit only. Note that these sequences are in the YUV 4:2:0 format. Our iWave++ is applied on the Y, U, V channels separately. PSNR is also calculated on the three channels separately. For comparison, we use the latest reference software of H.265 and H.266, i.e., HM version 16.20[7] and VTM version 6.0,[8] respectively.

6. At the time of writing this paper, H.266/VVC is not officially published yet.
7. https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.20/
8. https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/tree/VTM-6.0/

Fig. 15 presents the results measured by Y-PSNR. Our lossy multi-model iWave++ outperforms H.265/HEVC significantly, and performs on par with H.266/VVC on Class D. On Classes B, C, E, our iWave++ is slightly worse than H.266/VVC. For quantitative comparison, Table 2 presents the BD-rate reduction ratios of our lossy multi-model iWave++ anchored on H.265 and H.266, respectively. Compared to H.265, our iWave++ achieves on average 17.17, 25.51, and 26.32 percent BD-rate reduction ratios on Y, U, V, respectively. Compared to H.266, our iWave++ incurs on average 6.27 percent BD-rate increase on Y, but achieves on average 8.42 and 5.30 percent BD-rate reduction ratios on U and V, respectively. These results demonstrate that our lossy multi-model iWave++ is one of the best schemes for image compression.

RGB Optimized iWave++. As mentioned before, our iWave++ is trained on the grayscale channel, and it is combined with a color transform (RGB to YUV here) to cope with RGB images. This design appears suboptimal in the sense of compression efficiency, but it has several advantages. First, the trained model can be directly used for grayscale images, RGB images, YUV 4:2:0 formatted images, or even multi-channel (e.g., hyperspectral) images. Second, it is believed that human vision system is less sensitive to chrominance than to luminance; thus, we may allocate more bitrate to luminance channel (e.g., Y) than to chrominance channel (e.g., U and V), which can be easily done as we process the channels separately. Third, the design makes a more fair comparison with traditional image compression schemes, such as JPEG, JPEG-2000, BPG, H.266/VVC, xvc, etc., since all of them work in the YUV color space by default.

Training in the RGB space is more straightforward for RGB image compression, which has been a common

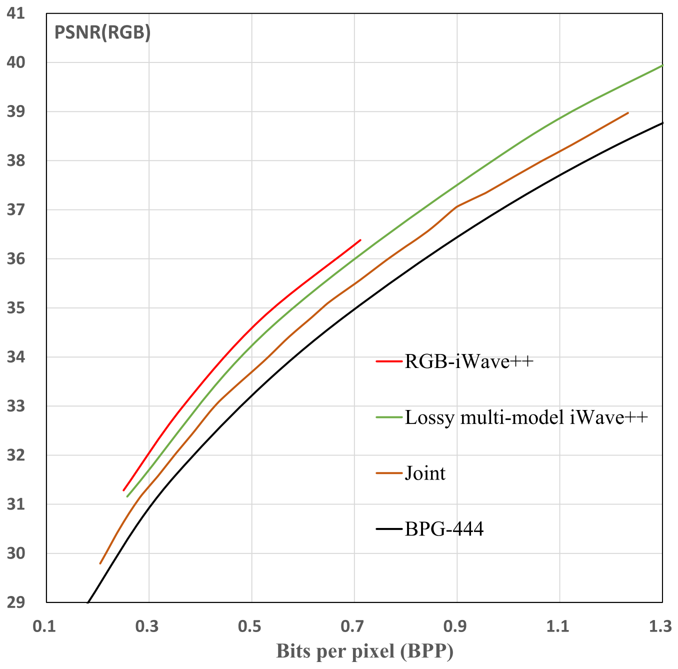


Fig. 16. Performance of “RGB-iWave++,” lossy multi-model iWave++ optimized in RGB color space, on the Kodak dataset. “Joint” refers to [6].

practice in end-to-end image compression. We also experiment with this option, but we keep the 2D transform as much as possible. Specifically, we use a 1×1× 3 (input channel)×3 (output channel) convolutional layer, acting as a *forward color transform*, as the first layer of the encoding network; subsequently, we use iWave forward transform on the three channels separately. At the decoder side, after iWave inverse transform, we use a 1×1 × 3× 3 convolutional layer as *inverse color transform*, and then use the dequantization module. The entire network is trained end-to-end. The results are shown in Fig. 16. It is indeed better than our previous design (trained on grayscale, used on YUV), which is understandable as the forward/inverse color transform is trained. We anticipate that 3D iWave, combining color transform and 2D spatial transform, may perform even better for RGB image compression. This is considered as our future work.

5.2.2 Lossy Variable Rate iWave++

The results of our lossy variable rate iWave++ are also shown in Figs. 10, 12, 13, 11, and 14. Lossy variable rate iWave++ performs worse than lossy multi-model iWave++, because the latter trains multiple models and each of which is optimized for a specific rate. Nonetheless, our lossy variable rate iWave++ is still very competitive. On the Kodak dataset, when evaluated with PSNR, it performs on par with *Joint* and outperforms all the other methods, as shown in Fig. 10. Anchored on BPG, it achieves on average 5.93 percent BD-rate reduction, slightly less than that of *Joint*, as shown in Fig. 12. On the Tecnick dataset, as shown in Fig. 11, it performs worse than *Joint*, but better than or comparable to all the other methods. Note that the compared end-to-end methods, *Joint* and *Variational*, use multiple models for multiple rates. On the contrary, our lossy variable rate iWave++ uses a single model for every rate. Its model storage overhead is minimal and its compression performance is comparable to state-of-the-art deep methods.

TABLE 3
Comparison of Different Lossless Compression Methods on ImageNet32 and ImageNet64 Test Sets (bits per pixel)

Method	ImageNet32	ImageNet64
Gated PixelCNN	11.49	10.71
Lossless iWave++	13.56	12.50
L3C	14.28	13.26
FLIF	15.24	13.62
WebP	15.84	13.92
JPEG-2000	19.05	15.21
PNG	19.26	17.22

5.2.3 Lossless iWave++

For lossless iWave++, we convert RGB images into the YCoCg color space as recommended in [49]. But our model is still trained on grayscale blocks as mentioned before.

We compare lossless iWave++ with four non-deep methods: PNG, JPEG-2000, WebP, FLIF, and two deep methods: L3C [31] and Gated PixelCNN [29]. Among them, FLIF is the best non-deep method. For comparison with L3C and Gated PixelCNN, we report the results on the ImageNet32 and ImageNet64 test sets in Table 3. Our lossless iWave++ surpasses all the compared methods except for Gated PixelCNN. Note that the context model in our lossless iWave++ has much less parameters than Gated PixelCNN. We also give the results on the Kodak dataset in Table 4, where the training dataset is DIV2K. Our lossless iWave++ is better than JPEG-2000 and WebP, slightly worse (∼ 3% bits increase) than FLIF.

5.2.4 Universal iWave++

Universal iWave++ is the most challenging configuration of iWave++, that is to use a single model for both lossless and lossy compression at different rates. To exclude the influence of color space conversion, we test our universal iWave++ as well as the compared methods (JPEG, JPEG-2000, BPG, WebP, FLIF) on grayscale images. We convert the Kodak images into grayscale for this test.

For lossy compression, Fig. 17 compares our universal iWave++ with JPEG, JPEG-2000, and BPG. Our universal iWave++ is better than JPEG and worse than BPG. Compared to JPEG-2000, our universal iWave++ performs worse at low bitrates but catches up and performs much better at high bitrates. For lossless compression, Table 5 presents the comparison results, which reveal that our universal iWave++ performs much better than JPEG-2000, WebP, and FLIF. In summary, our universal iWave++ is competitive for high-bit-rate to lossless image compression. Its performance

TABLE 4
Comparison of Different Lossless Compression Methods on the Kodak Dataset (bits per pixel)

Method	JPEG-2000	WebP	FLIF	iWave++
Kodak	9.58	9.56	8.68	8.97

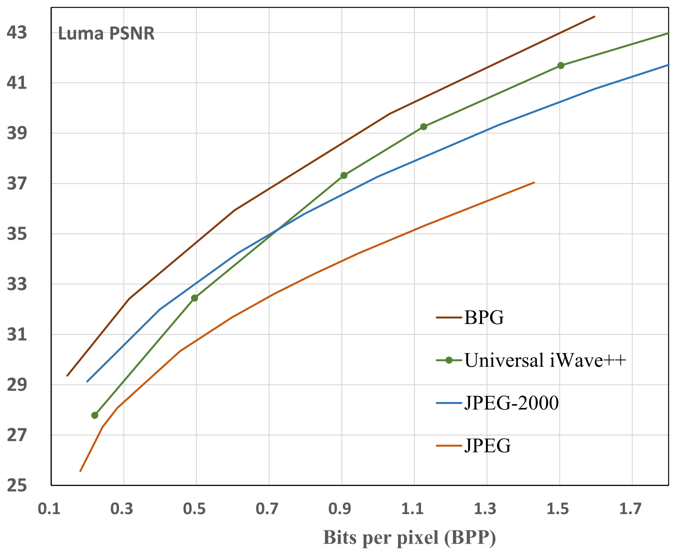


Fig. 17. Performance of the universal iWave++ on the grayscale Kodak dataset (converted to grayscale prior to compression). Shown are the average rate and the average luma PSNR of the 24 images.

TABLE 5
Comparison of Different Lossless Compression Methods on the Grayscale Kodak Dataset (bits per pixel)

Method	JPEG-2000	WebP	FLIF	Universal iWave++
Kodak (grayscale)	4.46	4.41	4.40	4.15

is limited at low bitrates since low-bit-rate compression is far away from lossless. It seems not easy to train a single compression model that is always optimal in the entire bitrate spectrum. Nonetheless, if we adjust the $\lambda_{\text{lossless}}$ in (20), we may obtain different results, which is an open question.

5.3 Ablation Studies

5.3.1 *On Entropy Coding*

We compare the proposed entropy coding tool with EBCOT, which is the recommended tool in JPEG-2000. First, we replace our tool entirely with EBCOT, whose results are given in Fig. 18 under the name "iWave++'s transform + EBCOT." It is much worse than iWave++, even worse than JPEG-2000, because the trained transform does not cooperate well with EBCOT. To make a fair comparison, we train the proposed entropy coding tool for the CDF 5/3 wavelet transform, which is the transform used in lossless JPEG-2000, and compare with EBCOT. Here, we conduct lossless compression of the grayscale Kodak dataset. Results are shown in Table 6, indicating that our tool achieves 11 percent bits saving (4.01 versus 4.46) than EBCOT.

We compare the heavy version and light version of the context model (shown in Figs. 7 and 8 respectively). Contrastive results are given in Fig. 18 (lossy) and Table 7 (lossless). The difference in compression efficiency is marginal in both lossy and lossless cases.

Moreover, we observe the effect of the long-term context extracting model by removing it from both the heavy version and the light version of the context model. The

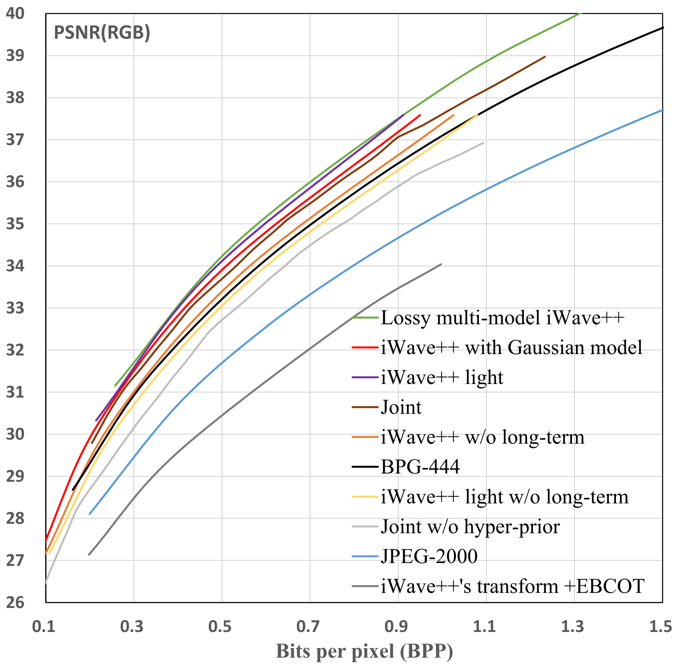


Fig. 18. Results of the ablation studies about the transform module and the entropy coding module. "iWave++ light" stands for lossy multi-model iWave++ using the light version of the context model (Fig. 8). "w/o long-term" is removing the long-term context model from either the heavy version or the light version of the context model (Figs. 7 and 8). "Joint" and "Joint w/o hyper-prior" are quoted from [6].

TABLE 6
Comparison of Different Lossless Compression Methods on the Grayscale Kodak Dataset (bits per pixel)

Method	Kodak (grayscale)
Lossless JPEG-2000	4.46
CDF 5/3 + Trained entropy coding	4.01
Lossless iWave++	3.99

TABLE 7
Comparison of Different Lossless Compression Methods on the Kodak Dataset

Context model	Using iWave?	Bits per pixel	Encoding Time (s)*
Heavy	N	8.84	1355
	Y	8.97 (+1.5%)	1201 (– 11.4%)
Light	N	9.27	1131
	Y	9.21 (– 0.6%)	993 (–12.2%)

* *For fair comparison, encoding is sequentially done for all the compared methods.*

corresponding results are shown in Fig. 18, which indicates that the long-term context extracting model boosts compression efficiency significantly. Note that without the long-term context, the light version suffers from more loss than the heavy version.

We replace the adopted probability model with a simple Gaussian model as the latter was used in *Joint* [6]. The corresponding results are shown in Fig. 18 under the name "iWave++ with Gaussian model." It is slightly worse than iWave++, demonstrating the effectiveness of the more complex probability model. Note that with the same probability model, iWave++ is still better than *Joint*.

5.3.2 On iWave

We compare the proposed revertible transform, iWave, with the non-revertible trained transform that was adopted in [6]. To make a fair comparison, we need to equip both transform tools with similar entropy coding tools: for iWave++, we use the light version of the context model without the long-term context; for the compared non-revertible transform, we use the autoregressive model, which is named *Joint without hyper-prior* in [6]. The two entropy coding tools are comparable because both are autoregressive and their parameter numbers are similar. Contrastive results are shown in Fig. 18 under the names of “iWave++ light w/o long-term” and “Joint w/o hyper-prior”, which demonstrates the advantage of the proposed transform than the nonlinear transform in [6].

We also compare iWave with the traditional wavelet transform, CDF 5/3. As shown in Table 6, when using the same (proposed) entropy coding, iWave is slightly better than CDF 5/3 for lossless compression (3.99 versus 4.01).

To better understand the interaction between iWave and the proposed entropy coding tool, we remove iWave and compress the pixels directly and losslessly using the entropy coding tool (both the heavy version and the light version). For pixel-domain coding, the long-term context model is discarded, resulting in pixel-wise autoregressive (PAR) model that is in spirit similar to PixelCNN [28]. The tested PAR models have much less parameters than PixelCNN [28] and PixelCNN++ [30] because we want to control the encoding/decoding complexity. The results are summarized in Table 7. When using the heavy context model with larger receptive field, pixel coding is better than transform coefficient coding. But when using the light context model with smaller receptive field, the benefit of transform is endorsed. This result can be attributed to the long-term context in iWave++ that leverages the multi-scale representation. The encoding time is also shown in Table 7. Using iWave is observed to reduce the encoding time because of the adopted multi-ary arithmetic coding: after iWave transform, high-frequency coefficients (LH, HL, HH subbands) usually have a smaller dynamic range than raw pixels, which reduces the arithmetic coding time.

5.3.3 On De-Quantization

When we remove the de-quantization module from iWave++, its compression performance drops significantly, as shown in Fig. 19. Nonetheless, even without the de-quantization module, iWave++ still performs on par with BPG especially at high bitrates. It is understandable that the de-quantization is more important at low bitrates because of more severe quantization loss.

As mentioned in Section 3.4, the de-quantization is proposed to address the problem of iWave++ that the revertible transform cannot compensate quantization error. This problem does not exist in other end-to-end schemes that do not enforce revertible transform. Indeed, in other end-to-end schemes, the decoding network already does some de-quantization, so it is less important to add another de-quantization module. To demonstrate this point, we augment a post-processing module to an existing end-to-end scheme, *HyperPrior-Mean* [6]. The post-processing module is structurally identical to our de-quantization module for fair comparison. The results are shown in Fig. 19. We also include into Fig. 19 the

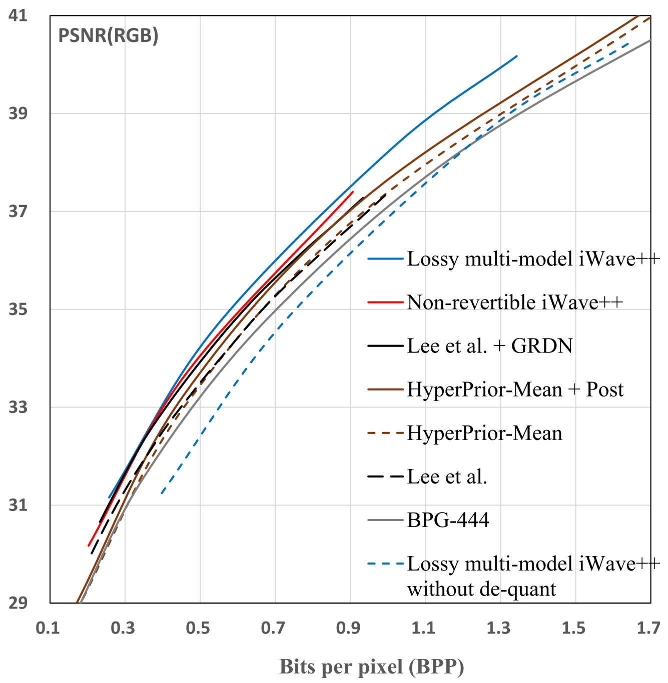


Fig. 19. Results of the ablation studies about the de-quantization module. “HyperPrior-Mean” refers to [6]. “HyperPrior-Mean + Post” is adding a post-processing module that is structurally identical to the de-quantization module. “Lee *et al.*” refers to [50]. “Lee *et al.* + GRDN” refers to [51].

results of *Lee et al.* [50] and *Lee et al. + GRDN* [51], where GRDN is a post-processing module and it is much more complex than our de-quantization module. It can be observed that the post-processing provides marginal gain for the state-of-the-art end-to-end schemes; the gain is far less than that obtained by the de-quantization in iWave++.

Another solution to compensate quantization error is to untie the binding of forward transform and inverse transform, i.e., forward transform and inverse transform are not enforced to share the same set of parameters. We experiment with this option, as we increase the complexity of the inverse transform network so that the total number of trainable parameters is comparable to that of the case with de-quantization. The results are given in Fig. 19 under the name “Non-revertible iWave++”. It has similar compression efficiency. However, we still recommend using the de-quantization to ensure the revertible transform.

5.4 Computational Complexity and Speed-Up

In this subsection we report the computational complexity of the lossy multi-model iWave++, and present a fast and light implementation. The timing results are measured on an NVIDIA GeForce RTX 2080 Ti GPU. The compared methods include JPEG-2000, BPG, xvc, *Variational*,[9] and *Lee et al*.[10]

The (slow) iWave++ has very high computational complexity, as shown in Table 8. Among its three modules: iWave, entropy coding, and de-quantization, we observe that entropy coding costs most of the time, which can also be evidenced by the results in Table 7.

9. We use the model for low bit rate compression. For high bit rate compression, the model has more parameters and longer running time.
10. We use the model for low bit rate compression. For high bit rate compression, the model has more parameters and longer running time.

TABLE 8
Average Running Time Per Image on the Kodak Dataset (in Seconds), and the Number of Training Parameters of Each Learning-Based Method

	JPEG-2000	BPG-444	xvc-444	Variational [2]	Lee *et al.* [50]	iWave++ (slow)	iWave++ (fast)
Avg. Enc. Time	0.14	0.26	30.58	1.10	10.28	1128.17	22.17
Avg. Dec. Time	0.08	0.11	0.04	0.93	40.68	1153.46	24.82
Number of Parameters	–	–	–	5.07M	6.47M	17.91M	1.29M

We propose two changes of the entropy coding model. First, we use the light version of the context model, shown in Fig. 8, which has been verified to incur little loss in Fig. 18. Second, to leverage the parallel computing of GPU, we design a block parallel coding strategy. When coding each subband, we divide the coefficients into 16×16 blocks, and process the blocks in parallel. Using this strategy, we omit the correlation among different blocks and thus suffer from a little loss in compression efficiency. Table 8 and Fig. 20 present the results of the fast iWave++ that integrates the two changes. The fast iWave++ reduces the encoding time by around 98 percent. Nonetheless, compared with the other methods, there is still large room for improvement.

In addition, the number of training parameters of each learning-based method is also listed in Table 8. The slow iWave++ has much more training parameters than *Variational* and *Lee et al.*, but the fast iWave++ has much less. It indicates that the "heavy" context model incurs a great amount of parameters. In summary, the entropy coding module seems having the highest impact on the computational complexity.

6 Conclusion

We have proposed iWave++ as a new end-to-end optimized image compression scheme, which consists of three important modules: iWave (trained wavelet-like transform), entropy coding, and de-quantization. iWave++ is unique among the existing end-to-end methods since the transform does not incur information loss. Accordingly, a single model supports both lossless and lossy compression, and simply achieves arbitrary compression ratio, making iWave++ a versatile scheme. Experimental results show that iWave++ achieves state-of-the-art compression performance for both lossy compression and lossless compression.

There are a number of possible improvements in the iWave ++ framework. First, we can explore advanced quantization methods for the coefficients, like vector quantization. Second, we can perform non-uniform quantization of the coefficients to realize bitrate allocation for rate-distortion optimization. Third, in universal iWave++, we can tradeoff the compression performance for lossy compression and for lossless compression. Besides the improvements, we plan to investigate end-to-end video compression with iWave or its 3D version.

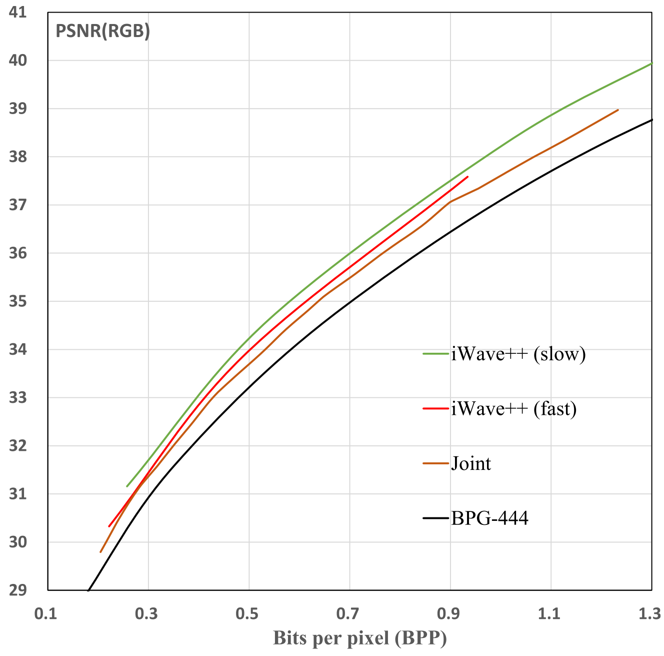


Fig. 20. Performance of the slow version and the fast version of lossy multi-model iWave++ on the Kodak dataset. Note that for the fast version, we not only use the light version of the context model (Fig. 8) but also use block parallel coding. "Joint" refers to [6].

Acknowledgments

This work was supported by the National Key Research and Development Program of China under Grant 2018YFA0701603, and the Natural Science Foundation of China under Grants 61772483 and 61931014.

References

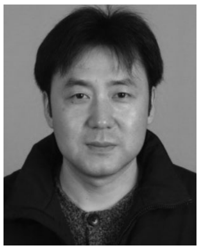
[1] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," 2016. [Online]. Available: https://arxiv.org/abs/1611.01704
[2] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," 2018. [Online]. Available: https://arXiv: 1802.01436
[3] N. Johnston *et al.*, "Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4385–4393.
[4] M. Li, W. Zuo, S. Gu, D. Zhao, and D. Zhang, "Learning convolutional networks for content-weighted image compression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 673–681.
[5] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. V. Gool, "Conditional probability models for deep image compression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4394–4402.
[6] D. Minnen, J. Ballé, and G. Toderici, "Joint autoregressive and hierarchical priors for learned image compression," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 10 794–10 803.
[7] O. Rippel and L. Bourdev, "Real-time adaptive image compression," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 2922–2930.
[8] L. Theis, W. Shi, A. Cunningham, and F. Huszár, "Lossy image compression with compressive autoencoders," 2017. [Online]. Available: https://arXiv: 1703.00395
[9] G. Toderici *et al.*, "Full resolution image compression with recurrent neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5306–5314.
[10] E. Agustsson, M. Tschannen, F. Mentzer, R. Timofte, and L. V. Gool, "Generative adversarial networks for extreme learned image compression," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 221–231.

[11] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
[12] Y. Choi, M. El-Khamy, and J. Lee, "Variable rate deep image compression with a conditional autoencoder," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 3146–3154.
[13] C. Christopoulos, A. Skodras, and T. Ebrahimi, "The JPEG2000 still image coding system: An overview," *IEEE Trans. Consum. Electron.*, vol. 46, no. 4, pp. 1103–1127, Nov. 2000.
[14] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.
[15] 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.
[16] H. Ma, D. Liu, R. Xiong, and F. Wu, "iWave: CNN-based wavelet-like transform for image compression," *IEEE Trans. Multimedia*, vol. 22, no. 7, pp. 1667–1679, Jul. 2020.
[17] W. Sweldens, "The lifting scheme: A custom-design construction of biorthogonal wavelets," *Appl. Comput. Harmon. Anal.*, vol. 3, no. 2, pp. 186–200, 1996.
[18] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps," *J. Fourier Anal. Appl.*, vol. 4, no. 3, pp. 247–269, 1998.
[19] G. Toderici *et al.*, "Variable rate image compression with recurrent neural networks," 2015. [Online]. Available: https://arXiv:1511.06085
[20] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimization of nonlinear transform codes for perceptual quality," in *Proc. Picture Coding Symp.*, 2016, pp. 1–5.
[21] H. Liu *et al.*, "Non-local attention optimized deep image compression," 2019. [Online]. Available: https://arXiv:1904.09757
[22] H. Liu, T. Chen, Q. Shen, and Z. Ma, "Practical stacked non-local attention modules for image compression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2019, pp. 1–4.
[23] J. Zhou, S. Wen, A. Nakagawa, K. Kazui, and Z. Tan, "Multi-scale and context-adaptive entropy model for image compression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2019, pp. 1–4.
[24] J. Lee *et al.*, "Extended end-to-end optimized image compression method based on a context-adaptive entropy model," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2019, pp. 1–4.
[25] E. Agustsson *et al.*, "Soft-to-hard vector quantization for end-to-end learning compressible representations," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1141–1151.
[26] C. Cai, L. Chen, X. Zhang, and Z. Gao, "Efficient variable rate image compression with multi-scale decomposition network," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 12, pp. 3687–3700, Dec. 2019.
[27] T. Dumas, A. Roumy, and C. Guillemot, "Autoencoder based image compression: Can the learning be quantization independent?" in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2018, pp. 1188–1192.
[28] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 1747–1756.
[29] A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu, "Conditional image generation with PixelCNN decoders," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 4790–4798.
[30] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma, "PixelCNN++: Improving the PixelCNN with discretized logistic mixture likelihood and other modifications," 2017. [Online]. Available: https://arXiv:1701.05517
[31] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. Van Gool, "Practical full resolution learned lossless image compression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 10 629–10 638.
[32] J. Ho, E. Lohn, and P. Abbeel, "Compression with flows via local bits-back coding," 2019. [Online]. Available: https://arXiv: 1905.08500
[33] E. Hoogeboom, J. W. Peters, R. V. D. Berg, and M. Welling, "Integer discrete flows and lossless compression," 2019. [Online]. Available: https://arXiv:1905.07376
[34] R. L. Claypoole, G. M. Davis, W. Sweldens, and R. G. Baraniuk, "Nonlinear wavelet transforms for image coding via lifting," *IEEE Trans. Image Process.*, vol. 12, no. 12, pp. 1449–1459, Dec. 2003.
[35] J.-H. Jacobsen, A. Smeulders, and E. Oyallon, "i-RevNet: Deep invertible networks," 2018. [Online]. Available: https://arXiv:1802.07088
[36] J. M. Shapiro, "An embedded hierarchical image coder using zerotrees of wavelet coefficients," in *Proc. Data Compression Conf.*, 1993, pp. 214–223.
[37] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 243–250, Jun. 1996.
[38] D. Taubman, "High performance scalable image compression with EBCOT," *IEEE Trans. Image Process.*, vol. 9, no. 7, pp. 1158–1170, Jul. 2000.
[39] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," 2015. [Online]. Available: https://arXiv:1506.00019
[40] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," 2014. [Online]. Available: https://arXiv:1409.2329
[41] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[42] M. Li, S. Gu, D. Zhang, and W. Zuo, "Enlarging context with low cost: Efficient arithmetic coding with trimmed convolution," 2018. [Online]. Available: https://arXiv:1801.04662
[43] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2017, pp. 136–144.
[44] Y. Dai, D. Liu, and F. Wu, "A convolutional neural network approach for post-processing in HEVC intra coding," in *Proc. Int. Conf. Multimedia Model.*, 2017, pp. 28–39.
[45] J. Kang, S. Kim, and K. M. Lee, "Multi-modal/multi-scale convolutional neural network based in-loop filter design for next generation video codec," in *Proc. IEEE Int. Conf. Image Process.*, 2017, pp. 26–30.
[46] E. Agustsson and R. Timofte, "NTIRE 2017 challenge on single image super-resolution: Dataset and study," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2017, pp. 126–135.
[47] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. [Online]. Available: https://arXiv:1412.6980
[48] N. Asuni and A. Giachetti, "TESTIMAGES: A large data archive for display and algorithm testing," *J. Graph. Tools*, vol. 17, no. 4, pp. 113–125, 2013.
[49] J. Sneyers and P. Wuille, "FLIF: Free lossless image format based on maniac compression," in *Proc. IEEE Int. Conf. Image Process.*, 2016, pp. 66–70.
[50] J. Lee, S. Cho, and S.-K. Beack, "Context-adaptive entropy model for end-to-end optimized image compression," 2018. [Online]. Available: https://arXiv:1809.10452
[51] J. Lee, S. Cho, and M. Kim, "An end-to-end joint learning scheme of image compression and quality enhancement with improved entropy minimization," 2019. [Online]. Available: https://arXiv:1912.12817

Haichuan Ma received the BS degree in electrical engineering from Xidian University, Xi'an, China, in 2017. He is currently working toward the PhD degree from the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei, China. His research interests include image/video coding, signal processing, and machine learning.

Dong Liu (Senior Member, IEEE) received the BS and PhD degrees in electrical engineering from the University of Science and Technology of China (USTC), Hefei, China, in 2004 and 2009, respectively. He was a member of research staff with Nokia Research Center, Beijing, China, from 2009 to 2012. He joined USTC as an associate professor, in 2012. His research interests include image and video coding, multimedia signal processing, and multimedia data mining. He has authored or coauthored more than 100 papers in international journals and conferences. He has 16 granted patents. He has one technical proposal adopted by AVS. He received the 2009 IEEE Transactions on Circuits and Systems for Video Technology Best Paper Award and the VCIP 2016 Best 10 percent Paper Award. He and his team were winners of several technical challenges held in ICCV 2019, ACM MM 2018, ECCV 2018, CVPR 2018, and ICME 2016. He is a senior member of the CCF and CSIG, and an elected member of MSA-TC of IEEE CAS Society. He served as a registration co-chair for ICME 2019 and a symposium co-chair for WCSP 2014.

Ning Yan received the BS degree in information engineering from the China University of Mining and Technology, Xuzhou, China, in 2015. He is currently working toward the PhD degree in electronic engineering and information science at the University of Science and Technology of China, Hefei, China. His research interests include video coding/processing and machine learning.



Houqiang Li (Senior Member, IEEE) received the BS, MEng, and PhD degrees in electronic engineering from the University of Science and Technology of China (USTC), Hefei, China, in 1992, 1997, and 2000, respectively. He is currently a professor at the Department of Electronic Engineering and Information Science of USTC. His research interests include video coding and communication, multimedia search, image/video analysis. He has authored and co-authored more than 100 papers in journals and conferences. He served as an associate editor of the IEEE Transactions on Circuits and Systems for Video Technology from 2010 to 2013, and has been with the editorial board of the *Journal of Multimedia* since 2009. He was the recipient of the Best Paper Award for Visual Communications and Image Processing (VCIP), in 2012, the recipient of the Best Paper Award for International Conference on Internet Multimedia Computing and Service (ICIMCS), in 2012, the recipient of the Best Paper Award for the International Conference on Mobile and Ubiquitous Multimedia from ACM (ACM MUM), in 2011, and a senior author of the Best Student Paper of the 5th International Mobile Multimedia Communications Conference (MobiMedia), in 2009.

Feng Wu (Fellow, IEEE) received the BS degree in electrical engineering from Xidian University, China, in 1992, and the MS and PhD degrees in computer science from the Harbin Institute of Technology, China, in 1996 and 1999, respectively. He is currently a professor and the assistant to the president at the University of Science and Technology of China, Hefei, China. Previously, he was a principle researcher and research manager with Microsoft Research Asia, Beijing, China. His research interests include various aspects of video technology and artificial intelligence. He has authored or co-authored more than 120 journal papers (including several dozens of IEEE Transactions papers) and top conference papers on MOBICOM, SIGIR, CVPR and ACM MM. He has 80 granted US patents. His 15 techniques have been adopted into international video coding standards. He serves or had served as the deputy editor-in-chief for IEEE Transactions on Circuits and System for Video Technology and as an associate editor for IEEE Transactions on Image Processing and IEEE Transactions on Multimedia. He also serves as general chair in ICME 2019, TPC chair in MMSP 2011, VCIP 2010 and PCM 2009. He received the best paper awards in IEEE TCSVT 2009, VCIP 2016, PCM 2008, and VCIP 2007, and best associate editor Award of IEEE Transactions on Image Processing in 2018.