WeConvene: Learned Image Compression with Wavelet-Domain Convolution and Entropy Model

Haisheng Fu[1], Jie Liang[1], Zhenman Fang[1], Jingning Han[2], Feng Liang[3], and Guohe Zhang[3]

[1] School of Engineering Science, Simon Fraser University, Canada
[2] Google LLC, USA
[3] School of Microelectronics, Xi'an Jiaotong University, China

Abstract. Recently learned image compression (LIC) has achieved great progress and even outperformed the traditional approach using DCT or discrete wavelet transform (DWT). However, LIC mainly reduces spatial redundancy in the autoencoder networks and entropy coding, but has not fully removed the frequency-domain correlation explicitly as in DCT or DWT. To leverage the best of both worlds, we propose a surprisingly simple but efficient WeConvene framework, which introduces the DWT to both the convolution layers and entropy coding of CNN-based LIC. First, in both the core and hyperprior autoencoder networks, we propose a Wavelet-domain Convolution (WeConv) module, which performs convolution after DWT, and then converts the data back to spatial domain via inverse DWT. This module is used at selected layers in a CNN network to reduce the frequency-domain correlation explicitly and make the signal sparser in DWT domain. We also propose a Wavelet-domain Channel-wise Auto-Regressive entropy Model (WeChARM), where the output latent representations from the encoder network are first transformed by the DWT, before applying quantization and entropy coding, as in the traditional paradigm. Moreover, the entropy coding is split into two steps. We first code all low-frequency DWT coefficients, and then use them as prior to code high-frequency coefficients. The channel-wise entropy coding is further used in each step. By combining WeConv and WeChARM, the proposed WeConvene scheme achieves superior R-D performance compared to other state-of-the-art LIC methods as well as the latest H.266/VVC. For the Kodak dataset and the baseline network with -0.4% BD-Rate saving over H.266/VVC, introducing WeConv with the simplest Haar transform improves the saving to -4.7%. This is quite impressive given the simplicity of the Haar transform. Enabling Haar-based WeChARM entropy coding further boosts the saving to -8.2%. When the Haar transform is replaced by the 5/3 or 9/7 wavelet, the overall saving becomes -9.4% and -9.8% respectively. The standalone WeConv layer can also be used in many other computer vision tasks beyond image/video compression. The source code is available at https://github.com/fengyurenpingsheng/WeConvene.

Keywords: Learned Image Compression · Wavelet Transform · Learning in Wavelet Domain · Wavelet-domain Convolution · Wavelet-domain Entropy Coding

1 Introduction

In the last few years, learned image compression (LIC) methods have quickly outperformed the traditional approaches in both subjective and objective metrics. Linear transform such as the discrete cosine transform (DCT) and discrete wavelet transform (DWT) is a key component in the traditional paradigm, followed by quantization and entropy coding in the transform domain. In LIC, the linear transform is replaced by deep learning-based neural networks, which can be more powerful than linear transform in learning the compact latent representations of the images.

Earlier LIC designs were mainly based on convolutional neural networks (CNN) [5, 13, 16, 17, 20, 23, 24, 30]. Recently the transformer network has been introduced [28, 32, 38], which can achieve better rate-distortion (R-D) performance, but transformer-based schemes are more difficult to train and have higher requirements on the GPU. These neural networks are also used in the entropy coding part to learn the distributions of the latent representations. As a result, many LIC schemes can get better performance than the traditional approaches, including intra coding in the latest H.266/VVC video coding standard.

Despite its great success, a major limitation of the state-of-the-art LIC schemes is that they do not explicitly remove the frequency-domain redundancy of the latent representations. Although there are some efforts in introducing transform-domain processing to the LIC [1–3, 10, 14, 19, 25, 29, 37], their performances are not satisfactory.

In this paper, we propose a surprisingly simple but efficient way of using DWT in both the autoencoder network and entropy coding parts of the LIC framework, and demonstrate that DWT can indeed significantly improve the performance in the learned image compression paradigm, as expected from the experience in the traditional approach.

Our contributions are summarized as follows:

- We propose an effective, low-cost, modular, and plug-and-play WeConv layer, which embeds the convolution between DWT and IDWT, so that the module can still be one layer of a large CNN network. This allows it to enjoy the benefits of CNNs, and also improve the sparsity in the DWT domain. For the Kodak dataset and the baseline network with -0.4% BD rate saving over H.266/VVC, introducing WeConv with the simplest Haar transform can improve the saving to -4.7%, with negligible change of model size and running time.
- We propose a wavelet domain quantization and entropy coding, denoted as WeChARM, which can explicitly benefit from the improved sparsity given by the WeConv module. For the Kodak dataset, combining Haar-based WeConv and WeChARM entropy coding further boosts the saving to -8.2%, with moderate increase of model size and running time. When the Haar transform is replaced by the 5/3 or 9/7 wavelet, the overall saving can be improved to -9.4% and -9.8% respectively.

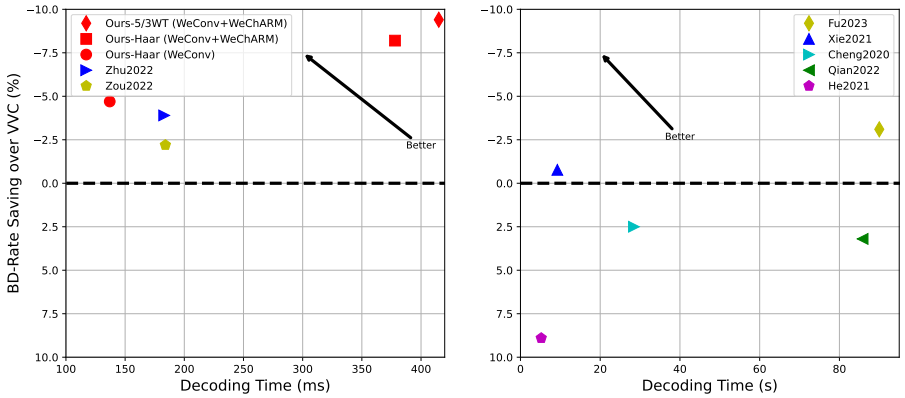


Fig. 1: The decoding time and BD-Rate reductions over H.266/VVC for different LIC schemes on the Kodak dataset.

– Since the proposed scheme is based on CNN, it is easier to train and has less requirements on GPU than transformer-based schemes. It also does not use other high complexity operators such as non-local modules. It therefore achieves a good trade-off between complexity and performance, as shown in Fig. 1, establishing our approach as the new state of the art in LIC.
– We show that with judicious design, the traditional wavelet transform can be used in LIC and achieve the state of the art. The scheme can be further improved. This opens up many future topics, and will bring in more attentions from the signal processing community. The proposed WeConv module can also be used in other applications beyond image compression.

2 Background and Related Work

2.1 Traditional Image Compression Methods

Traditional image and video codings, such as JPEG [35], JPEG 2000 [33], and H.264/H.265/H.266, extensively utilize linear transforms such as the Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT) to remove the redundancy in the frequency domain. The transformed data is then quantized to remove small coefficients in the frequency domain without introducing too much reconstruction error. The remaining redundancy is removed using entropy coding.

2.2 Representative LIC Methods

In the last few years, learned image compression (LIC) [11, 12, 15–17, 28, 32, 38] has witnessed remarkable advancements, and started to outperform traditional methods, including the latest H.266/VVC. In [6], the first end-to-end learned

image compression framework was proposed, which outperformed JPEG and JPEG2000 by using a novel architecture of convolutions and nonlinear activation functions. In [5], a variational autoencoder structure with a hyperprior is proposed for capturing spatial dependencies, achieving comparable performance to BPG (4:4:4). In [30], based on [5] and by combining autoregressive and hierarchical priors, the method was able to beat BPG (4:4:4) on both PSNR and MS-SSIM metrics.

Based on [30], some LIC methods [13,22,23,36] utilize serial context-adaptive models to achieve better performance than H.266/VVC. However, serial context models are time-consuming. To address this issue, a channel-wise autoregressive entropy model (ChARM) is introduced in [31] to avoid element-level serial processing. Furthermore, in [16], a spatial-channel contextual adaptive model is proposed to speed us the entropy coding without compromising the R-D performance. Similarly, in [17], a checkerboard context model (CCM) is developed to improve parallelism, but the R-D performance is reduced slightly compared to serial context model.

In [9], efficient residual network is proposed to extract more compact and efficient latent representation. Attention modules are also used, which has been adopted in some other schemes [14,22,39]. In [36], the invertible neural networks (INNs) are used to enhance overall performance.

Recently the transformer network has been introduced to LIC [28, 32, 38]. For example, in [28], the swin-transformer is combined with a ChARM model to enhance spatial dependency capture. However, transformer-based schemes are more difficult to train and have higher requirements on the GPU.

2.3 Efforts in Frequency-domain LIC

There have been some efforts in introducing frequency-domain processing to the LIC.

In [3], the authors applied "3-scale Daubechies-1" wavelets, and then introduced various CNN layers in the wavelet domain. The IDWT is only used at the end of the decoder. However, the performance was 5-6dB lower than JPEG, and 8-9 dB lower than JPEG2000. A similar idea was used in [19], where the 9/7 wavelet and the network in [5] are used, but its results are also not satisfactory.

In [1, 2, 10, 25], the octave convolution proposed in [8] is introduced in LIC, where the feature map in each layer is divided into a low-resolution part and a high-resolution part. The multi-resolution concept is similar to wavelet transform, but the learned filters for the two parts do not necessarily have high-pass or low-pass frequency responses.

In [14], the image is decomposed into a low-frequency (LF) part and the residual high-frequency (HF) part, via simple pooling and subtraction operators, similar to Laplacian pyramid. The two parts are processed separately and merged by a dual attention module. In [37], the idea is applied to a transformer-based LIC, where the heads in the multi-head self-attention module in the transformer are split into HF heads and LF heads, using pooling and subtraction.

In [29], the lifting structure in the wavelet transform is imposed by the neural network architecture, and the filters in each lifting step are learned via training, but it does not use the existing wavelet coefficients, such as 5/3 and 9/7 wavelets.

3 WeConvene: LIC with Wavelet-Domain Convolution and Entropy Model

In this section, we describe the entire architecture of the proposed method, the proposed new components of WeConv and WeChARM, the loss function, and the training of the system.

The proposed scheme is depicted in Fig. 2. As in other popular LIC methods, our system includes the core autoencoder g_a to extract the compact latent representations of the input image, the core decoder network g_s to reconstruct the image, the hyperprior networks h_a and h_s to encode and decode the side information that helps the entropy coding of the latents.

The input color image has dimension $W \times H \times 3$. The pixel values are normalized to the range of $[-1, 1]$. The encoder network g_a includes multiple layers of convolutions and leaky ReLU. Some layers are grouped into ResGroup modules, each includes three residual blocks, as shown in Fig. 2. Some layers use the proposed WeConv module, which includes the pooling or downsampling operation, and will be described in Sec. 3.1.

Another contribution of our scheme is to apply the DWT at the end of the core encoder network to convert the latent representations into the wavelet domain. This makes the coefficients sparser and can improve the subsequent quantization and entropy coding.

The wavelet-domain coefficients are then quantized. To reduce the bit rate, the entropy coding is divided into two steps. The LF quantized DWT subband $\hat{y}_L$ is first encoded, which is then used to encode/decode the three quantized HF DWT subbands $\hat{y}_H$.

As in [28, 31], we use the fast channel-wise entropy coding (ChARM) to encode the LF and HF coefficients, denoted by WeChARM (L) and WeChARM (H) in Fig. 2. The details of WeChARM will be explained in Sec. 3.2.

As in other LIC schemes, to improve the entropy coding performance, the hyperprior networks h_a and h_s are utilized to encode and decode additional prior information z for both y_L and y_H. The WeConv module is also used in the hyperprior encoding networks. The inverse WeConv (IWeConv) module with transposed convolution is used in the core decoding network and the hyperprior decoding network, as described in Sec. 3.1.

3.1 Wavelet-domain Convolution (WeConv) Module

Fig. 3 shows the details of the proposed WeConv and inverse WeConv modules. In this paper, we use WeConv when the size of the latent representation is changed (this might not be necessary in other applications). The input signal

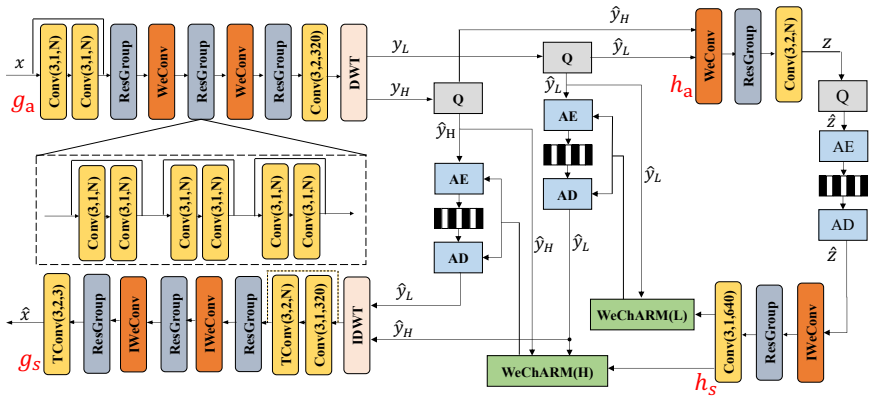


Fig. 2: The architecture of the proposed WeConvene scheme. Conv(3, s, n) represents a convolution layer with 3×3 kernel size, stride s, and n filters. TConv(3, s, n) is the transposed convolution. Dashed shortcut connections represent change of tensor size. *AE* and *AD* stand for Arithmetic Encoder and Arithmetic Decoder, respectively.

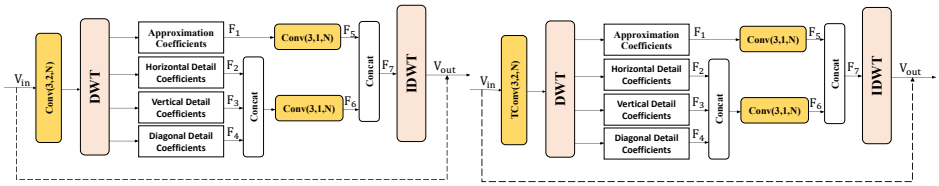


Fig. 3: (a) The architecture of forward WeConv network with downsampling; (b) The architecture of inverse WeConv (IWeConv) network with upsampling.

first passes through a convolutional layer, which also performs downsampling or upsampling, and then converted to the wavelet domain by the DWT operator.

In this paper, we use the 2×2 Haar transform, the 5/3 and 9/7 wavelets in JPEG 2000 as examples of the DWT. In Sec. 4, we will compare the performance of the three wavelets.

After the DWT, the coefficients in the LF subband, F_1, are filtered by one set of convolutions. The three HF subbands, F_2, F_3, F_4, are concatenated and filtered by another set of convolutions.

We then split the HF subbands to their original locations, and apply the inverse DWT to obtain the corresponding spatial-domain latent representations, which can be processed by the subsequent convolution layers as usual. The shortcut connection with different sizes as proposed in the ResNet is applied in the spatial domain.

The structure of the inverse WeConv module is similar to the forward WeConv, except that the transposed convolution is used to upsample the signal.

The Generalized Divisive Normalization (GDN) is used in WeConv and IWeConv modules, which has better performance than the Leakly ReLU [5].

The proposed WeConv module uses DWT to transform the input data into the wavelet domain, performs subband-based convolution, and then transforms the signal back to the time domain by IDWT. Therefore, it can be used as a standalone layer in CNN networks without drastically disrupting the typical signal distributions in CNNs, which could make it difficult to design a good network, as shown by the unsatisfactory performance in [3, 19], where the entire CNNs are performed in the wavelet domain.

3.2 Wavelet-domain Channel-Wise Auto-Regressive Entropy Model (WeChARM)

In this part, we explain the details of the two WeChARM modules in Fig. 2 to encode the LF and HF components y_L and y_H in the wavelet domain, as shown in Fig. 4 and Fig. 5.

The channel-wise auto-regressive entropy (ChARM) module was first introduced in [31]. In [28], a Swin-transformer-based attention mechanism (SWAtten) is used. It also reduces the number of slices in [31] from 10 to 5 to improve the trade-off between running speed and R-D performance.

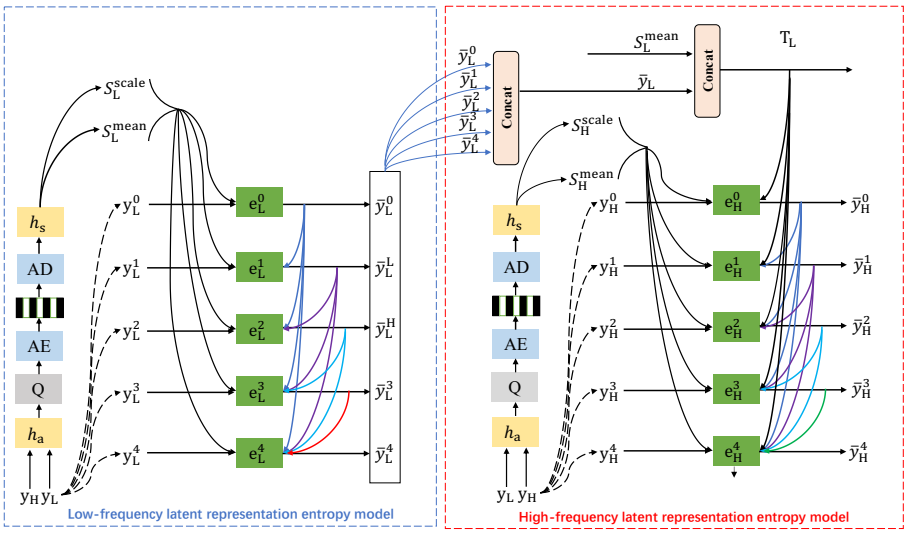


Fig. 4: The details of the proposed WeChARM modules for LF and HF subbands.

In this paper, we apply the ChARM model in [28] with 5 slices to encode both the LF and HF components y_L and y_H in the wavelet domain, as shown in Fig. 4. Each slice includes 64 channels. Since our probability modeling is learned

in the wavelet domain, it is sparser and more efficient than in the spatial domain. As a result, we found that we can remove the time-consuming SWAtten module in [28] without affecting the R-D performance.

The five LF slices y_L^i are encoded sequentially by five slice coding networks e_L^i $(i = 0, ..., 4)$, with the help of the side information S_L^{scale} and S_L^{mean} from hyperprior network (y_L^i is assumed to follow a Gaussian distribution), as well as outputs from the preceding slices to reduce inter-slice redundancy.

After the LF components are coded, they are used to code the five HF slices y_H^i via five networks e_H^i.

Fig. 5 illustrates the details of the slice coding network e_H^i. The network structure of e_L^i is similar to e_H^i, except that there is no prior information from y_L.

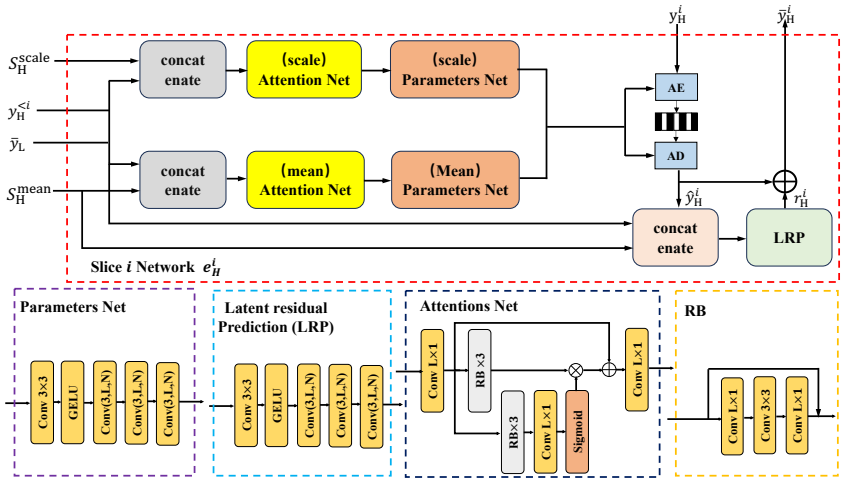


Fig. 5: The slice coding network e_H^i for the HF entropy coding in Fig. 4.

3.3 Loss Function

Our loss function is to optimize the R-D performance of the system. Let R represent the expected bitstream length, and D denote the reconstruction error between the source and reconstructed images. The trade-off between rate and distortion is regulated by a Lagrange multiplier, λ. Consequently, the objective cost function is defined as follows:

$$\begin{aligned} L &= \lambda D(\boldsymbol{x}, \hat{\boldsymbol{x}}) + H(\hat{\boldsymbol{y}}_L) + H(\hat{\boldsymbol{y}}_H) + H(\hat{\boldsymbol{z}}), \\ H(\hat{\boldsymbol{z}}) &= E[-\log_2(P_{\hat{\boldsymbol{z}}}(\hat{\boldsymbol{z}}))], \\ H(\hat{\boldsymbol{y}}_L) &= E[-\log_2(P_{\hat{\boldsymbol{y}}_L|\hat{\boldsymbol{z}}}(\hat{\boldsymbol{y}}_L|\hat{\boldsymbol{z}}))], \\ H(\hat{\boldsymbol{y}}_H) &= E[-\log_2(P_{\hat{\boldsymbol{y}}_H|\hat{\boldsymbol{y}}_L,\hat{\boldsymbol{z}}}(\hat{\boldsymbol{y}}_H|\hat{\boldsymbol{y}}_L,\hat{\boldsymbol{z}}))], \end{aligned} \tag{1}$$

where $D(\boldsymbol{x}, \hat{\boldsymbol{x}})$ is the distortion between the original image $\boldsymbol{x}$ and reconstructed image $\hat{\boldsymbol{x}}$. We utilize the mean squared error (MSE) and multi-scale structural similarity (MS-SSIM) respectively as our optimized metrics to train our networks. $H(\hat{\boldsymbol{y}}_L)$, $H(\hat{\boldsymbol{y}}_H)$ and $H(\hat{\boldsymbol{z}})$ represent the entropies of the LF, HF components and the hyperprior latent representations, respectively.

3.4 Model Training

The training images are obtained from the CLIC [34], LIU4K [27] and Coco datasets [26], and are resized to 2000×2000 pixels as part of the data augmentation process, which also includes rotation and scaling. We then obtain 160,000 training image patches with a resolution of 480×480 pixels.

Our models are optimized using MSE and MS-SSIM metrics respectively. For MSE optimization, the λ values are selected from the set of 0.0025, 0.0035, 0.0067, 0.013, 0.025, 0.05, each corresponding to a fixed bit rate, with the number of filters (N) for the latent features is set as 128 for all rates. For MS-SSIM metric, λ is set at 5, 8, 16, 32, and 64, with the filter number N remaining at 128. Each model is trained by 1.5×10^6 iterations using the Adam optimizer, with a batch size of 8 and an initial learning rate of 1×10^{-4}, which is trained every 100,000 iterations after the initial 750,000 iterations.

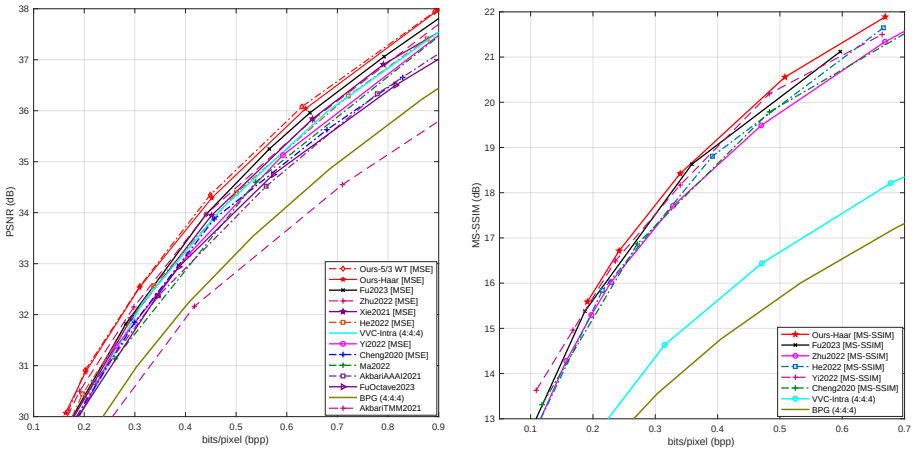
4 Experimental Results

This section evaluates the proposed method against some state-of-the-art LIC methods and traditional image codes, using both the Peak Signal-to-Noise Ratio (PSNR) and MS-SSIM metrics. The LIC methods include Fu2023 [13], FuOctave2023 [10], Zhu2022 [38], Yi2022 [32], He2022 [16], He2021 [17], Xie2021 [36], AkbariAAAI2021 [1], AkbariTMM202 [2], Cheng2020 [9], Minnen2020 [31], and Minnen2018 [30]. The traditional methods are H.266/VVC Intra (4:4:4), and H.265/BPG Intra (4:4:4).

Three popular test sets are selected, namely the Kodak PhotoCD test set [21] (24 images with 768×512 or 512×768 resolution), the Tecnick 100 test set [4] (100 images with 1200×1200 resolution), and the CLIC 2021 test set [34] (60 images with resolutions ranging from 751×500 to 2048×2048).

To ensure fair comparisons, we retain the Cheng2020 [9] method by increasing its number of filters N from 192 to 256 for scenarios requiring higher rates, thereby achieving better performance compared to the original results in [9] results. The results of other methods come from open-source codes or their original papers.

In the propose WeConvene method, we test three different wavelets: the 2×2 Haar transform, as well as the 5/3 wavelet and the 9/7 wavelet used in JPEG 2000. Symmetric extension is used at the boundary to avoid boundary artifact and improve the sparsity.



(a) **(b)**

Fig. 6: The average PSNR (a) and MS-SSIM (b) performances of different methods in the Kodak test set.

4.1 R-D Performance

Fig. 6 depicts the average R-D curves of different methods in all images of the Kodak dataset in terms of PSNR and MS-SSIM metrics. Among the other PSNR-optimized methods, Zhu2022 (MSE) [38] achieves the best performance when the bit rate is lower than 0.43 bpp. It is also better than H.266/VVC at all rates. When the bit rate is higher than 0.43 bpp, Fu2023 (MSE) [13] achieves the best performance.

Our proposed WeConvene method with the simple Haar transform consistently outperforms the best of Zhu2022 [38] and Fu2023 [13] by about 0.2 dB at all rates, and has a gain of more than 0.5 dB over VVC, especially at low rates. This is quite impressive given the simplicity of the Haar transform.

When the 5/3 wavelet is used, our performance can be further improved by up to 0.10 dB. The performance of 9/7 wavelet is very similar to 5/3 wavelet, as shown in Fig. 8 (b) later. Therefore the result of 9/7 wavelet is not shown in Fig. 6.

In the MS-SSIM metric in Fig. 6 (b), our method with the Haar transform also achieves the best performance. Better results can be expected using 5/3 and 9/7 wavelets.

Fig. 7 (a) reports the PSNR performances of different methods in the Tecnick 100 dataset. Among the PSNR-optimized methods, Xie2021 [36] achieves the best performance in other compared methods. Our method with Haar transform also outperforms Xie2021 [36] by about 0.2 dB at most rates, and achieves −9.46% BD-Rate reduction over VVC. Our method with 5/3 wavelet can further improve up to 0.1 dB.

Fig. 7 (b) compares the PSNR performances of different methods in the CLIC 2021 test set. Our method with Haar transform has even more gains over

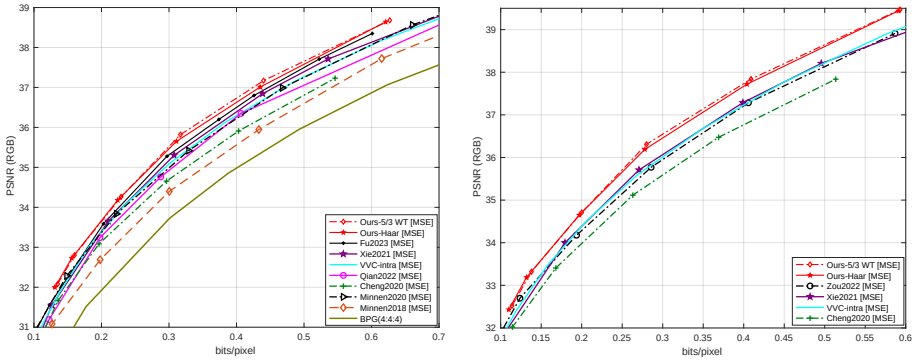


Fig. 7: (a) The average PSNR performances of different methods in the Tecnick dataset. (b) The average PSNR performances of different methods in the CLIC dataset.

Table 1: Comparisons of encoding/decoding time, BD-Rate reduction over VVC, and model sizes of the low bit rates and high bit rates for Kodak test set.

Methods	Enc. Time	Dec Time	BD-Rate	#Params
VVC	402.3s	0.61s	0.0	-
Cheng2020 [9]	27.6s	28.8s	2.6 %	50.80 MB
Hu2021 [18]	32.7s	77.8s	11.1 %	84.60 MB
He2021 [17]	20.4s	5.2s	8.9 %	46.60 MB
Xie2021 [36]	4.097s	9.250s	-0.8 %	128.86 MB
Zhu2022 [38]	0.269s	0.183s	-3.9 %	32.34 MB
Zou2022 [39]	0.163s	0.184s	-2.2%	99.86 MB
Qian2022 [32]	4.78s	85.82s	3.2 %	128.86 MB
Fu2023 [13]	420.6s	423.8s	-3.1%	-
Baseline	**0.109s**	**0.142s**	**-0.4%**	**52.22 MB**
Baseline+WeConv(Haar)	**0.110s**	**0.147s**	**-4.7%**	**58.41 MB**
WeConvene(Haar)	**0.352s**	**0.388s**	**-8.2%**	**107.15 MB**
WeConvene(5/3 WT)	**0.363s**	**0.415s**	**-9.4%**	**109.23 MB**
WeConvene(9/7 WT)	**0.386s**	**0.445s**	**-9.8%**	**113.46 MB**

Xie2021 [36], Zou2022 [39], and VVC, with up to 0.5 dB at high rates. Its BD-Rate reduction over VVC is -9.20%. Our method with 5/3 wavelet is also better than the Haar transform slightly.

4.2 Performance and Speed Trade-off

Table 1 compares the average encoding/decoding times, BD-rate reductions over VVC [7], and the number of model parameters (obtained by the PyTorch Flops Profiler tool) of various methods on the Kodak test set, using a NVIDIA Tesla V100 GPU with 12 GB memory, except for VVC, which only runs on CPU (a

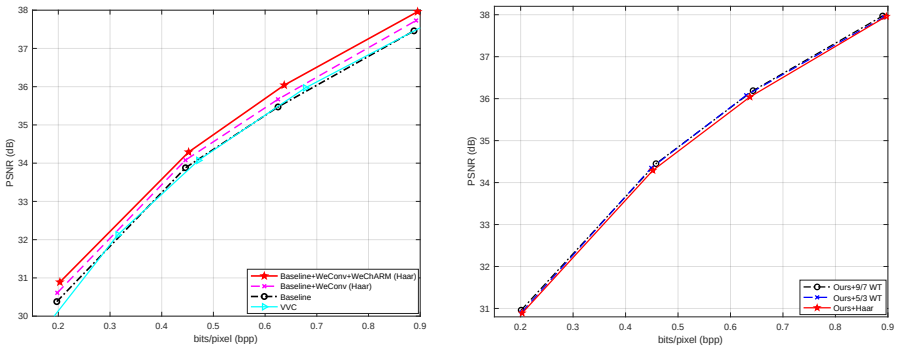


Fig. 8: (a) R-D performances of VVC and different configurations of our method for the Kodak dataset using the Haar transform. (b) R-D performances of VVC and different wavelets (Haar, 5/3, and 9/7 wavelets) in WeConvene for the Kodak dataset.

2.9GHz Intel Xeon Gold 6226R CPU is used). The number of parameters of [13] is not available, since it is written in TensorFlow, but it is shown in [13] that its model complexity is much higher than [9].

To study the contributions of WeConv and WeChARM separately in our method, we design a simplified baseline scheme for our method by removing the DWT/IDWT in WeConv and WeChARM, and only using one ChARM module in entropy coding. On top of the Baseline, we enable the WeConv and then the two-step WeChARMs. In each case, we retrain the entire system to get its best performance. Table 1 includes results of our Baseline, Baseline + WeConv (Haar), and the full WeConvene with Haar, 5/3, and 9/7 wavelets respectively.

The decoding time and BD-Rate reductions of some methods are also reported in Fig. 1 earlier.

The encoding and decoding times of learned methods [9, 13, 32, 36] are relatively slow because they employ sequential entropy models and cannot be accelerated by GPU. Some recent LIC approaches such as [38, 39] are much faster, by using GPU-friendly parallelizable entropy models. Their R-D performances are also among the best.

The BD-Rate reduction of the proposed WeConvene scheme with 9/7 wavelet is 5.9% better than [38], and −9.8% better than VVC, making our method the new state of the art. Our model complexity with different wavelets is only 8−14% higher than [39]. The encoding/decoding time is about twice of [39]. This is mainly because we use two sequential WeChARM modules in the entropy coding part, but our method is still significantly faster than many other LIC methods.

4.3 Contributions of Different Modules in WeConvene

Table 1 includes the performances of our Baseline and Baseline + WeConv (Haar). Both of them are faster than [38, 39]. The BD-rate reduction of the

Baseline over VVC is only −0.4%. Enabling WeConv with the Haar transform almost does not increase the encoding/decoding time, but it can achieve an impressive −4.7% BD-rate reduction over VVC, which is already better than other LIC methods in the table. The model complexity is only increased by about 10% compared to the Baseline.

Fig. 8 (a) compares the R-D curves of VVC and different configurations of our scheme on the Kodak dataset using the Haar transform. It can be seen that our baseline achieves similar performance to VVC. When WeConv is enabled, the performance is improve by about 0.2 dB at all rates. When WeChARM is also enabled, another gain of 0.2 dB can be achieved.

4.4 Contributions of Different Wavelets

In this experiment, we replace the Haar wavelet with the 9/7 and 5/3 wavelet. Other configurations remain the same. The experimental results are shown in Fig. 8 (b). The 5/3 wavelet improves performance by about 0.05-0.1 dB at the same bit rate compared to the Haar. The 9/7 wavelet has almost the same performance as the 5/3 wavelet. The reason is that the input sizes to the WeConv modules are not very large in this paper.

4.5 Comparison of Different Channel Slices in WeChARM

Table 2: The performance of different channel slices

Groups	Bit rate	PSNR	MS-SSIM	Enc. time	Dec. time	#Params
5	0.162	30.12 dB	12.78 dB	0.352 ms	0.388 ms	107.15 MB
10	0.167	30.16 dB	12.82 dB	0.424 ms	0.491 ms	179.09 MB
5	0.894	37.96 dB	20.53 dB	0.352 ms	0.388 ms	107.15 MB
10	0.9023	38.01 dB	20.57 dB	0.424 ms	0.491 ms	179.09 MB

Table 2 studies the impact of the number of channel slices in the channel-wise entropy coding when the Haar transform is used. Results with 5 and 10 slices at low rate and high rate are reported.

It can be observed that at both low rate and high rate, when the latent representations are divided into 10 slices instead of 5 slices, the R-D performance only increases slightly. On the other hand, the model size increases about 67%, and the encoding/decoding time increases 20-25%. This is because when there are too many slices, the number of channels is smaller in each slice, making it less efficient to reduce the redundancy. Moreover, since the slices need to be coded sequentially, the encoding/decoding time is also increased. Therefore we choose to use 5 slices in WeChARM.

5 Conclusions

This paper introduces a simple but efficient approach to use wavelet transform in both the convolution layers and entropy coding of the learned image compression (LIC). It makes the latent representations sparser in wavelet domain, which helps to achieve better R-D performance.

For the Kodak dataset and the baseline network with -0.4% BD-Rate saving over H.266/VVC, introducing WeConv with the simplest Haar transform improves the saving to -4.7%. This is quite impressive given the simplicity of the Haar transform. Enabling Haar-based WeChARM entropy coding further boosts the saving to -8.2%. When the Haar transform is replaced by the 5/3 or 9/7 wavelet, the overall saving becomes -9.4% and -9.8% respectively. The complexity of the scheme is also significantly lower than most LIC methods.

The framework in this paper opens up many future research topics, and allows the rich theories and results in the wavelet community to be introduced to learned image/video coding. For example, multiple levels of wavelet transforms can also be employed. Another possible approach is to use different wavelets in different layers, e.g., longer wavelets when the input size is larger, and shorter wavelets when the input is smaller.

In addition, as a standalone convolution layer module, the WeConv can also be used in many other computer vision tasks beyond image/video compression.

6 Acknowledgments

This work was supported by the Natural Sciences and Engineering Research Council of Canada (RGPIN-2020-04525), Google Chrome University Research Program, NSERC Discovery Grant RGPIN-2019-04613, DGECR-2019-00120, Alliance Grant ALLRP-552042-2020; CFI John R. Evans Leaders Fund, the National Natural Science Foundation of China (No. 61474093), and Industrial Field Project - Key Industrial Innovation Chain (Group) of Shaanxi Province (2022ZDLGY06-02), .

References

1. Akbari, M., Liang, J., Han, J., Tu, C.: Learned bi-resolution image coding using generalized octave convolutions. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 35, pp. 6592–6599 (Feb 2021) 2, 4, 9
2. Akbari, M., Liang, J., Han, J., Tu, C.: Learned multi-resolution variable-rate image compression with octave-based residual blocks. IEEE Transactions on Multimedia (2021) 2, 4, 9
3. Akyazi, P., Ebrahimi, T.: Learning-based image compression using convolutional autoencoder and wavelet decomposition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (June 2019) 2, 4, 7

4. Asuni, N., Giachetti, A.: TESTIMAGES: a Large-scale Archive for Testing Visual Devices and Basic Image Processing Algorithms. The Eurographics Association (2014). https://doi.org/10.2312/stag.20141242 9
5. Ballé, J., Minnen, D., Singh, S., Hwang, S.J., Johnston, N.: Variational image compression with a scale hyperprior. In: International Conference on Learning Representations. pp. 1–23 (2018) 2, 4, 7
6. Ballé, J., Laparra, V., Simoncelli, E.P.: End-to-end optimized image compression. In: International Conference on Learning Representations (2017) 3
7. Bjontegaard, G.: Calculation of average PSNR differences between RD curves (2001), VCEG-M33 11
8. Chen, Y., Fan, H., Xu, B., Yan, Z., Kalantidis, Y., Rohrbach, M., Yan, S., Feng, J.: Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 3435–3444 (2019) 4
9. Cheng, Z., Sun, H., Takeuchi, M., Katto, J.: Learned image compression with discretized gaussian mixture likelihoods and attention modules. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 7939–7948 (2020) 4, 9, 11, 12
10. Fu, H., Liang, F.: Learned image compression with generalized octave convolution and cross-resolution parameter estimation. Signal Processing **202**, 108778 (2023) 2, 4, 9
11. Fu, H., Liang, F., Liang, J., Fang, Z., Zhang, G., Han, J.: Efficient learned image compression with selective kernel residual module and channel-wise causal context model. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 4040–4044 (2024) 3
12. Fu, H., Liang, F., Liang, J., Li, B., Zhang, G., Han, J.: Asymmetric learned image compression with multi-scale residual block, importance scaling, and post-quantization filtering. IEEE Transactions on Circuits and Systems for Video Technology **33**(8), 4309–4321 (2023) 3
13. Fu, H., Liang, F., Lin, J., Li, B., Akbari, M., Liang, J., Zhang, G., Liu, D., Tu, C., Han, J.: Learned image compression with gaussian-laplacian-logistic mixture model and concatenated residual modules. IEEE Transactions on Image Processing **32**, 2063–2076 (2023) 2, 4, 9, 10, 11, 12
14. Gao, G., You, P., Pan, R., Han, S., Zhang, Y., Dai, Y., Lee, H.: Neural image compression via attentional multi-scale back projection and frequency decomposition. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 14677–14686 (2021) 2, 4
15. Guo, Z., Zhang, Z., Feng, R., Chen, Z.: Causal contextual prediction for learned image compression. IEEE Transactions on Circuits and Systems for Video Technology **32**(4), 2329–2341 (2022). https://doi.org/10.1109/TCSVT.2021.3089491 3
16. He, D., Yang, Z., Peng, W., Ma, R., Qin, H., Wang, Y.: Elic: Efficient learned image compression with unevenly grouped space-channel contextual adaptive coding. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5718–5727 (June 2022) 2, 3, 4, 9
17. He, D., Zheng, Y., Sun, B., Wang, Y., Qin, H.: Checkerboard context model for efficient learned image compression. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 14771–14780 (June 2021) 2, 3, 4, 9, 11

18. Hu, Y., Yang, W., Ma, Z., Liu, J.: Learning end-to-end lossy image compression: A benchmark. IEEE Transactions on Pattern Analysis and Machine Intelligence pp. 1–1 (2021). https://doi.org/10.1109/TPAMI.2021.3065339 11
19. Iliopoulou, S., Tsinganos, P., Ampeliotis, D., Skodras, A.: Learned image compression with wavelet preprocessing for low bit rates. In: 2023 24th International Conference on Digital Signal Processing (DSP). pp. 1–5 (2023) 2, 4, 7
20. Jiang, W., Yang, J., Zhai, Y., Ning, P., Gao, F., Wang, R.: Mlic: Multi-reference entropy model for learned image compression. In: Proceedings of the 31st ACM International Conference on Multimedia. pp. 7618–7627 (2023). https://doi.org/10.1145/3581783.3611694 2
21. Kodak, E.: Kodak lossless true color image suite (photocd pcd0992) (1993), http://r0k.us/graphics/kodak/ 9
22. Koyuncu, A.B., Gao, H., Boev, A., Gaikov, G., Alshina, E., Steinbach, E.: Contextformer: A transformer with spatio-channel attention for context modeling in learned image compression. In: Computer Vision – ECCV 2022. pp. 447–463 (2022) 4
23. Lee, J., Cho, S., Kim, M.: Joint autoregressive and hierarchical priors for learned image compression. arXiv:1912.12817 (2020) 2, 4
24. Lee, J., Cho, S., Beack, S.K.: Context-adaptive entropy model for end-to-end optimized image compression. In: International Conference on Learning Representations (2019) 2
25. Lin, J., Akbari, M., Fu, H., Zhang, Q., Wang, S., Liang, J., Liu, D., Liang, F., Zhang, G., Tu, C.: Variable-rate multi-frequency image compression using modulated generalized octave convolution. In: 2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP). pp. 1–6 (2020) 2, 4
26. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Computer Vision – ECCV 2014. pp. 740–755 (2014) 9
27. Liu, J., Liu, D., Yang, W., Xia, S., Zhang, X., Dai, Y.: A comprehensive benchmark for single image compression artifact reduction. IEEE Transactions on Image Processing **29**, 7845–7860 (2020) 9
28. Liu, J., Sun, H., Katto, J.: Learned image compression with mixed transformer-cnn architectures. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 14388–14397 (June 2023) 2, 3, 4, 5, 7, 8
29. Ma, H., Liu, D., Yan, N., Li, H., Wu, F.: End-to-end optimized versatile image compression with wavelet-like transform. IEEE Transactions on Pattern Analysis and Machine Intelligence **44**(3), 1247–1263 (2022). https://doi.org/10.1109/TPAMI.2020.3026003 2, 5
30. Minnen, D., Ballé, J., Toderici, G.D.: Joint autoregressive and hierarchical priors for learned image compression. In: Advances in Neural Information Processing Systems. pp. 10794–10803 (2018) 2, 4, 9
31. Minnen, D., Singh, S.: Channel-wise autoregressive entropy models for learned image compression. In: 2020 IEEE International Conference on Image Processing (ICIP). pp. 3339–3343 (2020) 4, 5, 7, 9
32. Qian, Y., Lin, M., Sun, X., Tan, Z., Jin, R.: Entroformer: A transformer-based entropy model for learned image compression. In: International Conference on Learning Representations (May 2022) 2, 3, 4, 9, 11, 12
33. Taubman, D.S., Marcellin, M.W.: JPEG2000: image compression fundamentals, standards, and practice. Kluwer Academic Publishers (2002) 3

34. Toderici, G., Timofte, R., Balle, J., Agustsson, E., Johnston, N., Mentzer, F.: 2021 workshop and challenge on learned image compression (clic). http://www.compression.cc 9
35. Wallace, G.K.: The jpeg still picture compression standard. IEEE Transactions on Consumer Electronics **38**(1), 18–34 (1992) 3
36. Xie, Y., Cheng, K.L., Chen, Q.: Enhanced invertible encoding for learned image compression. In: Proceedings of the ACM International Conference on Multimedia. pp. 162–170 (2021) 4, 9, 10, 11, 12
37. Zafari, A., Khoshkhahtinat, A., Mehta, P., Ebrahimi Saadabadi, M.S., Akyash, M., Nasrabadi, N.M.: Frequency disentangled features in neural image compression. In: 2023 IEEE International Conference on Image Processing (ICIP). pp. 2815–2819 (2023) 2, 4
38. Zhu, Y., Yang, Y., Cohen, T.: Transformer-based transform coding. In: International Conference on Learning Representations (2022) 2, 3, 4, 9, 10, 11, 12
39. Zou, R., Song, C., Zhang, Z.: The devil is in the details: Window-based attention for image compression. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 17492–17501 (June 2022) 4, 11, 12