Learned Image Compression with Mixed Transformer-CNN Architectures

Jinming Liu[14]   Heming Sun[23*]   Jiro Katto[12]
[1]Department of Computer Science and Communication Engineering, Waseda University, Tokyo, Japan
[2]Waseda Research Institute for Science and Engineering, Waseda University, Tokyo, Japan
[3]JST PRESTO, 4-1-8 Honcho, Kawaguchi, Saitama, Japan   [4] Shanghai Jiao Tong University, Shanghai, China
{jmliu@toki., hemingsun@aoni., katto@}waseda.jp

Abstract

Learned image compression (LIC) methods have exhibited promising progress and superior rate-distortion performance compared with classical image compression standards. Most existing LIC methods are Convolutional Neural Networks-based (CNN-based) or Transformer-based, which have different advantages. Exploiting both advantages is a point worth exploring, which has two challenges: 1) how to effectively fuse the two methods? 2) how to achieve higher performance with a suitable complexity? In this paper, we propose an efficient parallel Transformer-CNN Mixture (TCM) block with a controllable complexity to incorporate the local modeling ability of CNN and the non-local modeling ability of transformers to improve the overall architecture of image compression models. Besides, inspired by the recent progress of entropy estimation models and attention modules, we propose a channel-wise entropy model with parameter-efficient swin-transformer-based attention (SWAtten) modules by using channel squeezing. Experimental results demonstrate our proposed method achieves state-of-the-art rate-distortion performances on three different resolution datasets (i.e., Kodak, Tecnick, CLIC Professional Validation) compared to existing LIC methods. The code is at `https://github.com/jmliu206/LIC_TCM`.

1. Introduction

Image compression is a crucial topic in the field of image processing. With the rapidly increasing image data, lossy image compression plays an important role in storing and transmitting efficiently. In the passing decades, there were many classical standards, including JPEG [31], WebP [11], and VVC [29], which contain three steps: transform, quantization, and entropy coding, have achieved impressive Rate-Distortion (RD) performance. On the other hand, different from the classical standards, end-to-end learned image compression (LIC) is optimized as a whole. Some very recent LIC works [5, 13, 32, 34, 36, 37] have outperformed VVC which is the best classical image and video coding standards at present, on both Peak signal-to-noise ratio (PSNR) and Multi-Scale Structural Similarity (MS-SSIM). This suggests that LIC has great potential for next-generation image compression techniques.

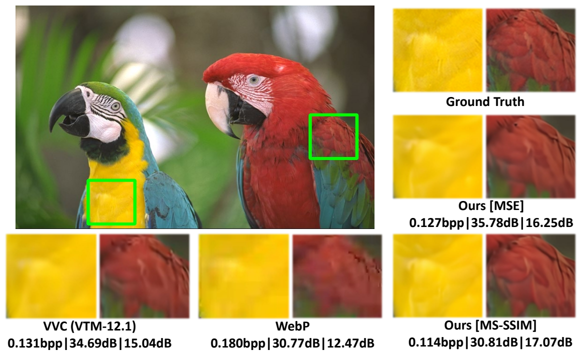


Figure 1. Visualization of decompressed images of $kodim23$ from Kodak dataset based on different methods (The subfigure is titled as "Method|Bit rate|PSNR|MS-SSIM").

Most LIC methods are CNN-based methods [6, 10, 20, 21, 33] using the variational auto-encoder (VAE) which is proposed by Ballé *et al*. [3]. With the development of vision transformers [9,22] recently, some vision transformer-based LIC methods [23, 36, 37] are also investigated. For CNN-based example, Cheng *et al*. [6] proposed a residual block-based image compression model. For transformer-based example, Zou *et al*. [37] tried a swin-transformer-based image compression model. These two kinds of methods have different advantages. CNN has the ability of local modeling, while transformers have the ability to model non-local information. It is still worth exploring whether the advantages of these two methods can be effectively combined with a suitable complexity. In our method, we try to efficiently incorporate both advantages of CNN and transformers by proposing an efficient parallel Transformer-CNN Mixture (TCM) block under a controllable complexity to improve

*Heming Sun is the corresponding author.

the RD performance of LIC.

In addition to the type of the neural network, the design of entropy model is also an important technique in LIC. The most common way is to introduce extra latent variables as hyper-prior to convert the probability model of compact coding-symbols to a joint model [3]. On that basis, many methods spring up. Minnen *et al.* [26] utilized the masked convolutional layer to capture the context information. Furthermore, they [27] proposed a parallel channel-wise auto-regressive entropy model by splitting the latent to 10 slices. The results of encoded slices can assist in the encoding of remaining slices in a pipeline manner.

Recently, many different attention modules [6, 21, 37] were designed and proposed to improve image compression. Attention modules can help the learned model pay more attention to complex regions. However, many of them are time-consuming, or can only capture local information [37]. At the same time, these attention modules are usually placed in both the main and the hyper-prior path of image compression network, which will further introduce large complexity because of a large input size for the main path. To overcome that problem, we try to move attention modules to the channel-wise entropy model which has $1/16$ input size compared with that of main path to reduce complexity. Nevertheless, if the above attention modules are directly added to the entropy model, a large number of parameters will be introduced. Therefore, we propose a parameter-efficient swin-transformer-based attention module (SWAtten) with channel squeezing for the channel-wise entropy model. At the same time, to avoid the latency caused by too many slices, we reduce the number of slices from 10 to 5 to achieve the balance between running speed and RD-performance. As Fig. 1 shows, our method can get pleasant results compared with other methods.

The contributions of this paper can be summarized as follows:

- We propose a LIC framework with parallel transformer-CNN mixture (TCM) blocks that efficiently incorporate the local modeling ability of CNN and the non-local modeling ability of transformers, while maintaining controllable complexity.
- We design a channel-wise auto-regressive entropy model by proposing a parameter-efficient swin-transformer-based attention (SWAtten) module with channel squeezing.
- Extensive experiments demonstrate that our approach achieves state-of-the-art (SOTA) performance on three datasets (i.e., Kodak, Tecnick, and CLIC datasets) with different resolutions. The method outperforms VVC (VTM-12.1) by 12.30%, 13.71%, 11.85% in Bjøntegaard-delta-rate (BD-rate) [4] on Kodak, Tecnick, and CLIC datasets, respectively.

2. Related Work

2.1. Learned end-to-end Image Compression

2.1.1 CNN-based Models

In the past decade, learned image compression has made significant progress and demonstrated impressive performance. Ballé *et al.* [2] firstly proposed an end-to-end learned CNN-based image compression model. Then they proposed a VAE architecture and introduced a hyper-prior to improve image compression in [3]. Furthermore, a local context model was utilized to improve the entropy model of image compression in [26]. In addition to that, a causal context was proposed by using global context information in [12]. Since the context model is time-consuming, He *et al.* [14] designed a checkerboard context model to achieve parallel computing, while Minnen *et al.* [27] used channel-wise context to accelerate the computing. Apart from improving the entropy model, some works attempt to adopt different types of convolutional neural networks to enhance image compression, such as Cheng *et al.* [6] who developed a residual network. Chen *et al.* [5] introduced octave residual networks into image compression models, and Xie *et al.* [34] used invertible neural networks (INNs) to improve performance.

2.1.2 Transformer-based Models

With the rapid development of vision transformers, transformers show impressive performance on not only high-level vision tasks, such as, image classification [22, 30], but also some low-level vision tasks, such as, image restoration [19], and image denoising [35]. Motivated by those works, some transformer-based LIC models are also proposed recently. Some works [36, 37] tried to construct a swin-transformer-based LIC model. Qian *et al.* [28] used a ViT [9] to help the entropy model capture global context information. Koyuncu *et al.* [18] utilized a sliding window to reduce the complexity of ViT in entropy models. Kim *et al.* [15] proposed an Information Transformer to get both global and local dependencies.

2.2. Attention Modules

Attention modules try to help the learned models focus on important regions to obtain more details. Many attention modules designed for image compression significantly improve the RD-performance. Liu *et al.* [21] firstly introduced a non-local attention module into image compression. Because of the non-local block, this attention module is time-consuming. Therefore, Cheng *et al.* [6] removed this non-local block and proposed a local attention module to accelerate the computing. Furthermore, Zou *et al.* [37] adopted a window-based attention module to improve image compression.

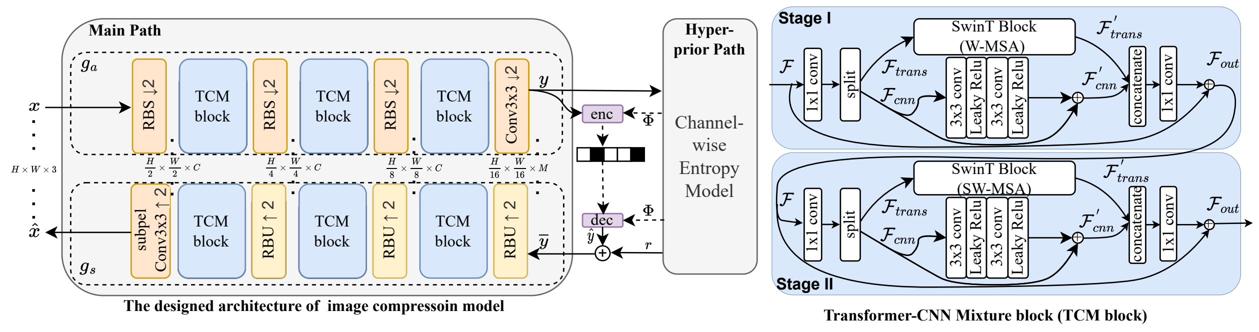


Figure 2. The overall framework of our method (left) and the proposed TCM block (right), enc and dec contain the processes of quantizing and range coding. ↑ 2 or ↓ 2 means that the feature size is enlarged/reduced by two times compared to the previous layer.

3. Proposed Method

3.1. Problem Formulation

As Fig.2 shows, the LIC models with a channel-wise entropy model [27] can be formulated by:

$$\begin{aligned} \boldsymbol{y} &= g_a(\boldsymbol{x}; \boldsymbol{\phi}) \\ \hat{\boldsymbol{y}} &= Q(\boldsymbol{y} - \boldsymbol{\mu}) + \boldsymbol{\mu} \\ \hat{\boldsymbol{x}} &= g_s(\hat{\boldsymbol{y}}; \boldsymbol{\theta}) \end{aligned} \tag{1}$$

where $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ represent the raw images and decompressed images. By inputting $\boldsymbol{x}$ to the encoder g_a with learned parameters $\boldsymbol{\phi}$, we can get the latent representation $\boldsymbol{y}$ which is estimated to have a mean $\boldsymbol{\mu}$. To encode it, $\boldsymbol{y}$ is quantized to $\hat{\boldsymbol{y}}$ by quantization operator Q. According to previous works [13, 26] and discussion[1], we round and encode each $\lceil \boldsymbol{y} - \boldsymbol{\mu} \rfloor$ to the bitstream instead of $\lceil \boldsymbol{y} \rfloor$ and restore the coding-symbol $\hat{\boldsymbol{y}}$ as $\lceil \boldsymbol{y} - \boldsymbol{\mu} \rfloor + \boldsymbol{\mu}$, which can further benefit entropy models. Then, we use range coder to encode losslessly $\lceil \boldsymbol{y} - \boldsymbol{\mu} \rfloor$ which is modeled as a single Gaussian distribution with the variance $\boldsymbol{\sigma}$ to bitstreams, and transmit it to decoder g_s. In this overall pipline, $\boldsymbol{\Phi} = (\boldsymbol{\mu}, \boldsymbol{\sigma})$ in this paper is derived by a channel-wise entropy model as Equation 2, Fig. 2 and Fig. 4 show. The entropy model of [27] divides $\boldsymbol{y}$ to s even slices $\{\boldsymbol{y}_0, \boldsymbol{y}_1, ..., \boldsymbol{y}_{s-1}\}$ so that the encoded slices can help improve the encoding of subsequent slices, we formulate it as:

$$\begin{gathered} \boldsymbol{z} = h_a\left(\boldsymbol{y}; \boldsymbol{\phi}_h\right), \quad \hat{\boldsymbol{z}} = Q(\boldsymbol{z}) \\ \boldsymbol{\mathcal{F}}_{mean}, \boldsymbol{\mathcal{F}}_{scale} = h_s(\hat{\boldsymbol{z}}; \boldsymbol{\theta}_h) \\ \boldsymbol{r}_i, \boldsymbol{\Phi}_i = e_i(\boldsymbol{\mathcal{F}}_{mean}, \boldsymbol{\mathcal{F}}_{scale}, \overline{\boldsymbol{y}}_{<i}, \boldsymbol{y}_i),\ 0 <= i < s \\ \overline{\boldsymbol{y}}_i = \boldsymbol{r}_i + \hat{\boldsymbol{y}}_i \end{gathered} \tag{2}$$

where h_a denotes the hyper-prior encoder with parameters $\boldsymbol{\phi}_h$. It is used to get side information $\boldsymbol{z}$ to capture spatial dependencies among the elements of $\boldsymbol{y}$. A factorized density model $\boldsymbol{\psi}$ is used to encode quantized $\hat{\boldsymbol{z}}$ as $p_{\hat{\boldsymbol{z}}|\boldsymbol{\psi}}(\hat{\boldsymbol{z}} \mid \boldsymbol{\psi}) = \prod_j \left(p_{z_j|\boldsymbol{\psi}}(\boldsymbol{\psi}) * \mathcal{U}\left(-\frac{1}{2}, \frac{1}{2}\right)\right)(\hat{z}_j)$ where j specifies the position of each element or each signal. $\hat{\boldsymbol{z}}$ is then fed to the hyper-prior decoder h_s with parameters $\boldsymbol{\theta}_h$ for decoding to obtain two latent features $\boldsymbol{\mathcal{F}}_{mean}, \boldsymbol{\mathcal{F}}_{scale}$ which are used to be input to the following each slice network e_i. After that, each slice $\boldsymbol{y}_i$ is sequentially processed to get $\overline{\boldsymbol{y}}_{\boldsymbol{i}}$. During this process, encoded slices $\overline{\boldsymbol{y}}_{<i} = \{\overline{\boldsymbol{y}}_0, \overline{\boldsymbol{y}}_1, ..., \overline{\boldsymbol{y}}_{i-2}, \overline{\boldsymbol{y}}_{i-1}\}$ and current slice $\boldsymbol{y}_i$ are input to the slice network e_i to get the estimated distribution parameters $\boldsymbol{\Phi}_i = (\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i)$ to help generate bit-streams. Therefore, we can assume $p_{\hat{\boldsymbol{y}}|\hat{\boldsymbol{z}}}(\hat{\boldsymbol{y}} \mid \hat{\boldsymbol{z}}) \sim \mathcal{N}\left(\boldsymbol{\mu}, \boldsymbol{\sigma}^2\right)$. At the same time, the residual $\boldsymbol{r}_i$ is used to reduce the quantization errors $(\boldsymbol{y} - \hat{\boldsymbol{y}})$ which is introduced by quantization. Therefore, $\overline{\boldsymbol{y}}$ with less error is entered into the decoder g_s with learned parameters $\boldsymbol{\theta}$, instead of $\hat{\boldsymbol{y}}$ in Equation 1. At last, we can get the decompressed image $\hat{\boldsymbol{x}}$. Fig. 4 illustrates the detailed process of this channel-wise entropy model clearly.

In order to train the overall learned image compression model, we consider the problem as a Lagrangian multiplier-based rate-distortion optimization. The loss is defined as:

$$\begin{aligned} \mathcal{L} =& \mathcal{R}(\hat{\boldsymbol{y}}) + \mathcal{R}(\hat{\boldsymbol{z}}) + \lambda \cdot \mathcal{D}(\boldsymbol{x}, \hat{\boldsymbol{x}}) \\ =& \mathbb{E}\left[-\log_2\left(p_{\hat{\boldsymbol{y}}|\hat{\boldsymbol{z}}}(\hat{\boldsymbol{y}} \mid \hat{\boldsymbol{z}})\right)\right] + \mathbb{E}\left[-\log_2\left(p_{\hat{\boldsymbol{z}}|\boldsymbol{\psi}}(\hat{\boldsymbol{z}} \mid \boldsymbol{\psi})\right)\right] \\ &+ \lambda \cdot \mathcal{D}(\boldsymbol{x}, \hat{\boldsymbol{x}}) \end{aligned} \tag{3}$$

where λ controls the rate-distortion tradeoff. Different λ values are corresponding to different bit rates. $\mathcal{D}(\boldsymbol{x}, \hat{\boldsymbol{x}})$ denotes the distortion term which is calculated by Mean squared error (MSE) loss. $\mathcal{R}(\hat{\boldsymbol{y}}), \mathcal{R}(\hat{\boldsymbol{z}})$ denote the bit rates of latents $\hat{\boldsymbol{y}}$ and $\hat{\boldsymbol{z}}$.

3.2. Transformer-CNN Mixture Blocks

Image compression models based on CNN [6] have achieved excellent RD-performance. Besides, with the rapid development of vision transformers, some methods based on vision transformers [36] are also proposed and outperform CNN-based methods because transformers can

[1] https://groups.google.com/g/tensorflow-compression/c/LQtTAo6l26U/m/mxP-VWPdAgAJ

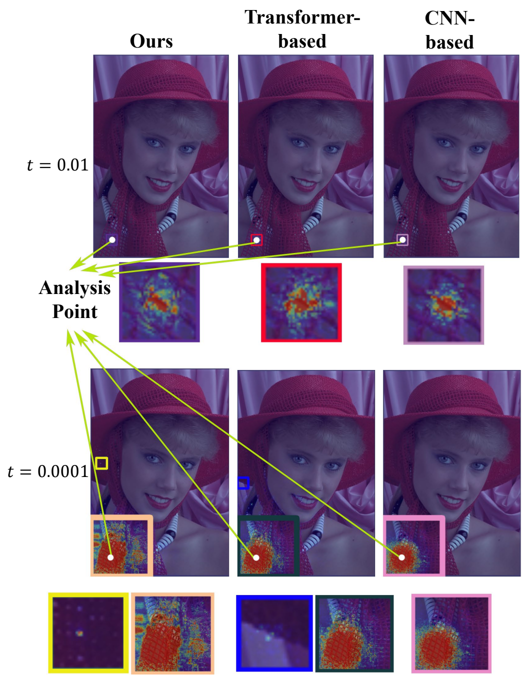


Figure 3. The effective receptive fields (ERF) calculated by different methods and clipped by different thresholds t. The two rows correspond to different thresholds while the three columns correspond to different methods including our proposal, transformer-based [37], and CNN-based [27] method.

capture non-local information. However, according to previous works [14, 37], even though non-local information can improve image compression, local information still has a significant impact on the performance of image compression. CNN can pay more attention to local patterns while transformers have the ability of non-local information. Therefore, we try to incorporate residual networks and Swin-Transformer (SwinT) blocks [22] to utilize both advantages of these two kinds of models. There are two challenges in this combination, the first is how to fuse two different features effectively, and the other is how to reduce the required complexity.

Here, an efficient parallel transformer-CNN mixture (TCM) block is proposed as Fig. 2 shows. We assume the input tensor as $\boldsymbol{\mathcal{F}}$ with size $C \times H_{\boldsymbol{\mathcal{F}}} \times W_{\boldsymbol{\mathcal{F}}}$. It is firstly input into a 1×1 convolutional layer whose output channels number is also C. Then we evenly split the tensor to two tensor $\boldsymbol{\mathcal{F}}_{cnn}$ and $\boldsymbol{\mathcal{F}}_{trans}$ with size $\frac{C}{2} \times H_{\boldsymbol{\mathcal{F}}} \times W_{\boldsymbol{\mathcal{F}}}$. This operation has two benefits. Firstly, it reduces the number of feature channels fed to subsequent CNN and transformers to decrease the model complexity. Secondly, local and non-local features can be processed independently and in parallel, which facilitates better feature extraction. After that, the tensor $\boldsymbol{\mathcal{F}}_{cnn}$ is sent to the residual network (Res) to get $\boldsymbol{\mathcal{F}}'_{cnn}$, while $\boldsymbol{\mathcal{F}}_{trans}$ is sent to the SwinT Blocks to get $\boldsymbol{\mathcal{F}}'_{trans}$. Then, we concatenate $\boldsymbol{\mathcal{F}}'_{trans}$ and $\boldsymbol{\mathcal{F}}'_{cnn}$ to get a tensor with size $C \times H_{\boldsymbol{\mathcal{F}}} \times W_{\boldsymbol{\mathcal{F}}}$. And then, the concatenated tensor is input to another 1×1 convolutional layer to fuse local and non-local features. At last, the skip connection between $\boldsymbol{\mathcal{F}}$ and the output is built to get $\boldsymbol{\mathcal{F}}_{out}$. In order to combine the residual network with the Swin-Transformer more effectively, we divide TCM into two stages with similar processes. In the transformer block of stage I, we use window-based multi-head self-attention (W-MSA). In stage II, we use shifted window-based multi-head self-attention (SW-MSA) in the transformer block. The benefit of this is that the residual networks are inserted into the common two consecutive Swin-transformer blocks, which can be more effective for feature fusion. Both two stages can be formulated as follows:

$$\begin{aligned}
&\boldsymbol{\mathcal{F}}_{cnn}, \boldsymbol{\mathcal{F}}_{trans} = Split(Conv1 \times 1(\boldsymbol{\mathcal{F}})) \\
&\boldsymbol{\mathcal{F}}'_{cnn}, \boldsymbol{\mathcal{F}}'_{trans} = Res(\boldsymbol{\mathcal{F}}_{cnn}), SwinT(\boldsymbol{\mathcal{F}}_{trans}) \\
&\boldsymbol{\mathcal{F}}_{out} = \boldsymbol{\mathcal{F}} + Conv1 \times 1(Cat(\boldsymbol{\mathcal{F}}'_{cnn}, \boldsymbol{\mathcal{F}}'_{trans}))
\end{aligned} \tag{4}$$

Based on the proposed TCM Block, the main path (g_a and g_s) is designed as Fig. 2 shows. The Residual Block with stride (RBS), Residual Block Upsampling (RBU) and subpixel conv3x3 are proposed by [6]. The detailed architectures of these three modules are reported in Supplementary. In our framework, except for the last layer of g_a and g_s, we attach a TCM block after each RBS/RBU to obtain non-local and local information. Also, we add TCM blocks into the hyper-prior path by re-designing the hyper-prior encoder h_a and hyper-prior decoder h_s as Fig. 4 shows.

To explore how the TCM block aggregates local and non-local information, we present effective receptive fields (ERF) [25] of our model. Besides, the effective receptive fields of the transformer-based model [37] and the CNN-based model [27] are used to make comparisons. ERF is defined as absolute gradients of a pixel in the output (i.e., $|\frac{d\hat{x}_p}{d\boldsymbol{x}}|$). Here, we calculate gradients of the analysis point $p = (70, 700)$ of $kodim04$ in Kodak dataset. In order to estimate the importance of information at different distances, we clip gradients with two thresholds t (i.e., 0.01, 0.0001) to visualize them. Note that the clip operation means we reduce the gradient values larger than the threshold to the threshold, and the gradient values smaller than the threshold remain unchanged. This can help our visualization has higher visibility precision. The visualization is shown in Fig. 3. As we can see, when $t = 0.01$, the red regions (high gradient values) of our TCM-based model and the CNN-based model are smaller than that of the transformer-based model. This suggests that our model pays more attention to neighbor regions, and has a similar local modeling ability of CNN. Meanwhile, when $t = 0.0001$, the ERF show our model and the transformer-based model can capture information at a long distance (As shown by the two small boxes in yellow and dark blue in Fig. 3, there is still a gradient

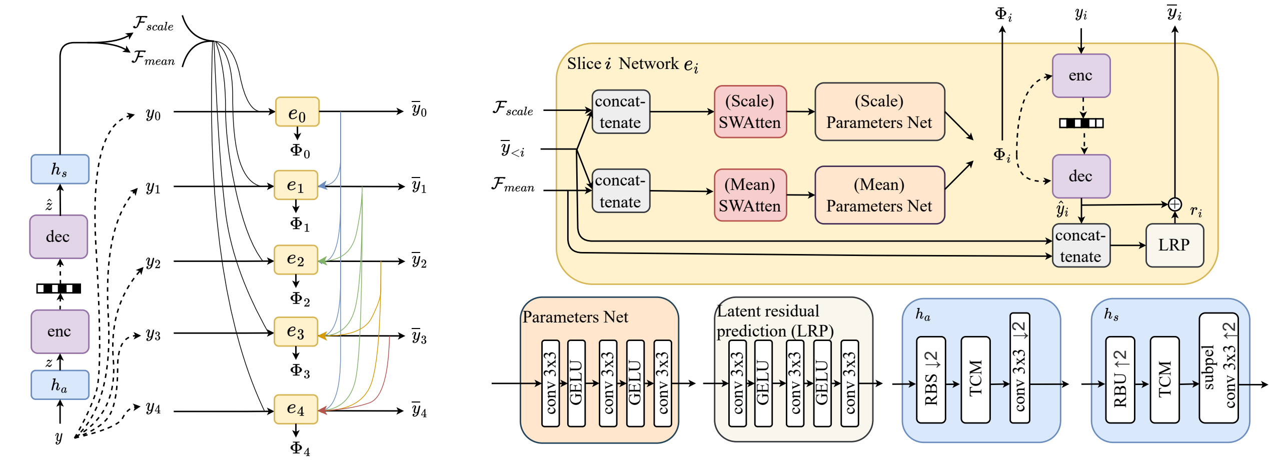


Figure 4. The proposed channel-wise entropy model. The encoded slice $\overline{\boldsymbol{y}}_{<i}$ can assist the encoding of the subsequent slice $\boldsymbol{y}_i$.

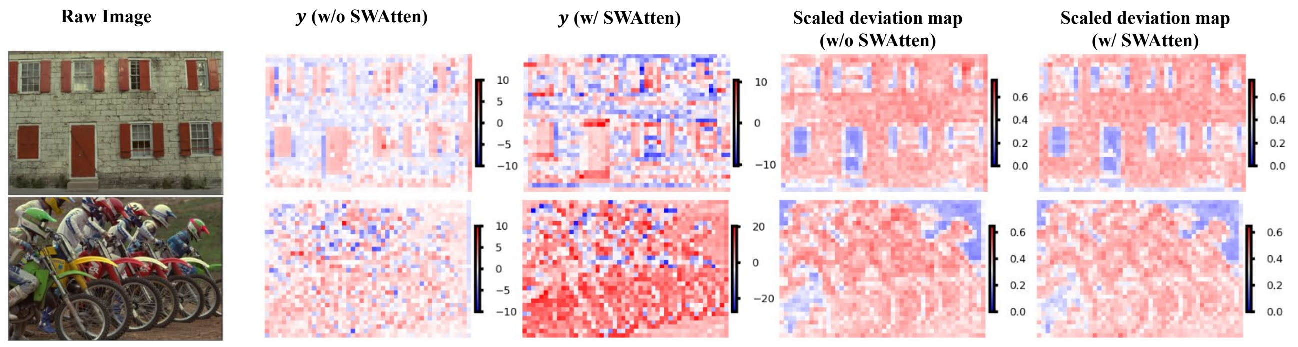


Figure 5. The scaled deviation map and the channel with the largest entropy of latent representations y, from the model without/with SWAtten. For $kodim01$ (top), the [PSNR@bitrates@ϵ_s] of the model without and with SWAtten are [29.61dB@0.503bpp@0.451] and [29.95dB@0.501bpp@0.389]. For $kodim05$ (bottom), they are [30.07dB@0.537bpp@0.422] and [31.03dB@0.534bpp@0.365].

return at a distance far from point p). This means that our model also has the long-distance modeling ability of transformers. It is also worth noting that at $t = 0.0001$, the CNN-based model exhibits a circular ERF, while the ERF of our model exhibit a shape closer to the context (a long strip shape like the background scarf). This shows that our model has better modeling ability compared to the CNN-based model.

3.3. Proposed Entropy Model

Motivated from [13, 27], we propose a channel-wise auto-regressive entropy model with a parameter-efficient swin-transformer-based attention module (SWAtten) by using channel squeezing. The framework is shown in Fig. 4.

3.3.1 SWAtten Module

Past works on attention have demonstrated their effectiveness for image compression. However, many of them are time-consuming, or only have the ability to capture local information. Different from these modules which are placed on both the main path (g_a and g_s) and hyper-prior path (h_a and h_s) of image compression, we design an attention module for the entropy model which has $1/16$ input size compared to the main path and can reduce much complexity. The designed parameter-efficient swin-Transformer based Attention (SWAtten) module is shown in Fig. 6. The swin-transformer block which can capture non-local information is added into the architecture, while the other residual blocks (RB) can get local information. Since the number of channels of features inputted to e_i accumulates with the increase of slice index i, the input channels of e_i can be expressed as:

$$M + i \times (M//s) \tag{5}$$

Where M is the number of channels of the latent variable y. The input channel number of e_9 can reach 608 when s is set as 10 and M is set as 320, which causes the model to be parameter-consuming. To achieve the balance between complexity and RD-performance, the total number of the slices s is reduced to 5 from 10 which is the common setting in [27]. At the same time, a channel squeeze operation is used to squeeze the input channels. In this paper, we squeeze the input channels of all slices to 128, i.e., let the output channel of the first 1×1 convolutional layer be 128. At last, an unsqueeze operation is utilized to unsqueeze the channels of output to the original number, i.e., let the output channel of the last 1x1 convolutional layer be $M + i \times (M/s)$.

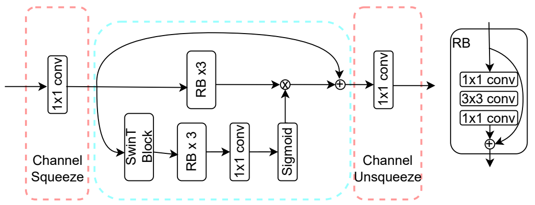


Figure 6. The proposed SWAtten module.

It should be noted that although both SWAtten and TCM are composed of transformers and CNNs, there are some differences between them. Firstly, according to [24], the hyper-prior path of the image compression network contains numerous redundant parameters, making it possible to use channel squeezing to greatly reduce the parameters without compromising performance. But the main path is sensitive to parameters, which means we cannot use such operations. Secondly, CNN in SWAtten is used not only for extracting local features but also for extracting attention maps, which differs from TCM. Lastly, the receptive field of the entropy model in SWAtten is large enough, eliminating the need for a two-stage fusion framework like TCM.

According to [34], the deviation between $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$ with size $M \times H_y \times W_y$ can be used to analyze the information loss in the process of compression, we can formulate the mean absolute pixel deviation ϵ as:

$$\begin{aligned}\epsilon &= \sum_{h=1}^{H_y}\sum_{w=1}^{W_y}\sum_{m=1}^{M} |\hat{y}_{h,w,m} - y_{h,w,m}| \\ &= \sum_{h=1}^{H_y}\sum_{w=1}^{W_y}\sum_{m=1}^{M} |Q(y_{h,w,m} - \mu_{h,w,m}) - (y_{h,w,m} - \mu_{h,w,m})|\end{aligned} \tag{6}$$

To compare the deviation among different models, it is unfair to directly compare the values ϵ because the $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$ in different models have different ranges. Since the deviation is relative to $\boldsymbol{y}$, a scaling factor $\gamma = \sum_{h=1}^{H_y}\sum_{w=1}^{W_y}\sum_{m=1}^{M} |y_{h,w,m}|$ is introduced to define a scaled mean absolute pixel deviation ϵ_s to better evaluate the information loss. ϵ_s can be formulated as:

$$\epsilon_s = \epsilon / \gamma \tag{7}$$

Fig. 5 shows the scaled deviation map and the channel with the largest entropy of $\boldsymbol{y}$ of $kodim01$ and $kodim05$ in Kodak dataset by using the model with SWAtten or not. Note that the deviation map is in shape (H_y, W_y) and each pixel is the mean of the absolute deviation along the channel dimension after scaling with γ. The ϵ_s for $kodim01$ and $kodim05$ can be reduced from 0.451 and 0.422 by using the model without SWAtten to 0.389 and 0.365 by using the model with SWAtten. It suggests that the model with SWAtten can have less information loss and get a higher quality decompressed image.

4. Experiments

4.1. Experimental Setup

4.1.1 Training Details

For training, we randomly choose 300k images of size larger than 256×256 from ImageNet [8], and randomly crop them with the size of 256×256 during the training process. We adopt Adam [16] with a batch size 8 to optimize the network. The initial learning rate is set as 1×10^{-4}. After 1.8M steps, the learning rate is reduced to 1×10^{-5} for the last 0.2M steps.

The model is optimized by RD-formula as Equation 3. Two kinds of quality metrics, i.e., mean square error (MSE) and MS-SSIM, are used to represent the distortion $\mathcal{D}$. When the model is optimized by MSE, the λ belongs to $\{0.0025, 0.0035, 0.0067, 0.0130, 0.0250, 0.0500\}$. When the model is optimized by MS-SSIM, the λ belongs to $\{3, 5, 8, 16, 36, 64\}$.

For swin-transformer blocks, window sizes are set as 8 in the main path (g_a and g_s), and 4 in the hyper-prior path (h_a and h_s). The channel number M of the latent $\boldsymbol{y}$ is set as 320, while that of $\boldsymbol{z}$ is set as 192, respectively. Other hyper-parameters in the entropy model follow the setting in [27]. We use RTX 3090 and Intel i9-10900K to complete the following experiments.

4.1.2 Evaluation

We test our method on three datasets, i.e., Kodak image set [17] with the image size of 768×512, old Tecnick test set[2] [1] with the image size of 1200×1200, CLIC professional validation dataset[3] [7] with 2k resolution. Both PSNR and MS-SSIM are used to measure the distortion, while bits per pixel (bpp) are used to evaluate bitrates.

4.1.3 Definition of Various Models

In the experiments, to explore the performance of our model with different complexities, we tested three different models (small, medium and large by setting different channel number $C = 128, 192, 256$ in the middle layers). The location of C is shown in Fig. 2. The number of slices s of the entropy model for all models with attention modules is reduced to 5 from 10 which is a common setting in [27]. More details are reported in Supplementary.

[2]https://sourceforge.net/projects/testimages/files/OLD/

[3]http://clic.compression.cc/2021/tasks/index.html

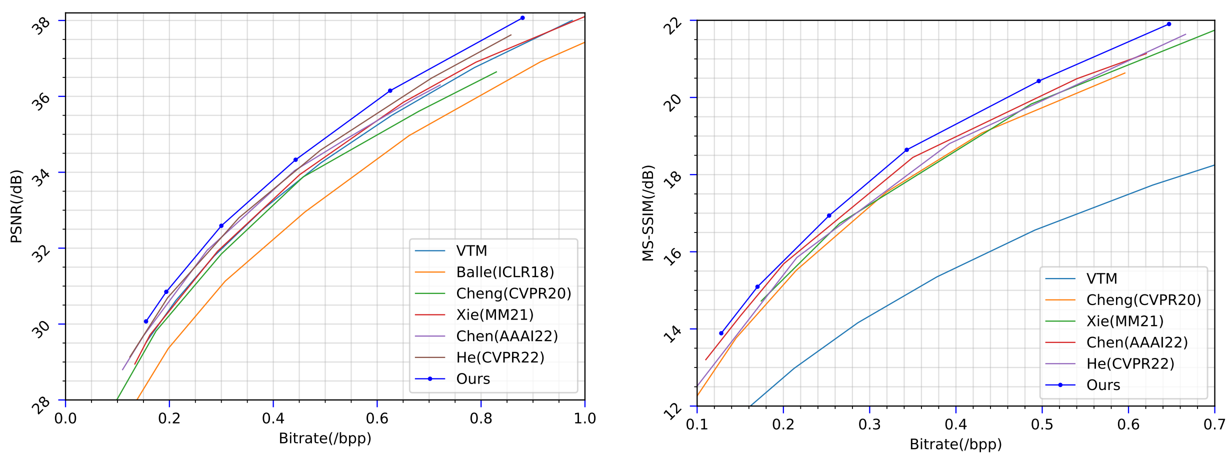


Figure 7. Performance evaluation on the Kodak dataset.

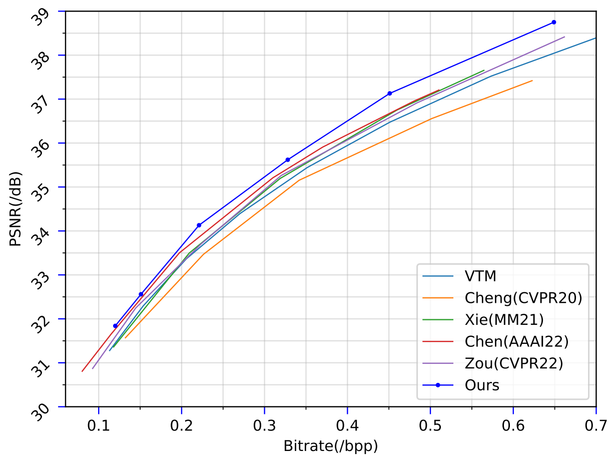


Figure 8. Performance evaluation on the CLIC Professional Validation dataset.

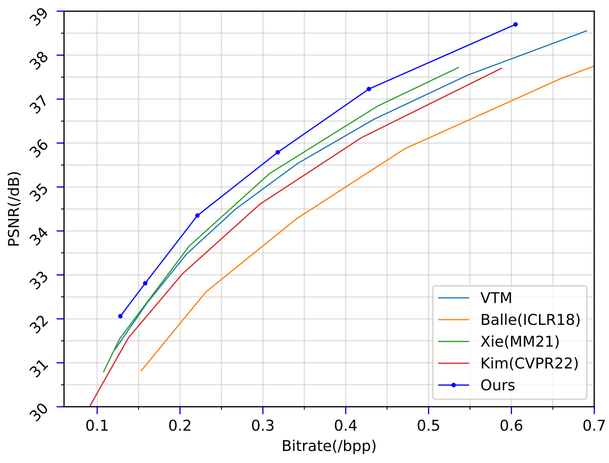


Figure 9. Performance evaluation on the Tecnick dataset.

4.2. Rate-Distortion Performance

We compare our large model with state-of-the-arts (SOTA) learned end-to-end image compression algorithms, including [3], [6], [34], [5], [15], [13], and [37]. The classical image compression codec, VVC [29] is also tested by using VTM12.1. The rate-distortion performance on Kodak dataset is shown in Fig. 7. Both PSNR and MS-SSIM are tested on Kodak to demonstrate the robustness of our method. Here, we convert MS-SSIM to $-10\log_{10}(1-\text{MS-SSIM})$ for clearer comparison. As we can see, at the same bitrate, we can improve up to about 0.4dB PSNR and 0.5dB MS-SSIM compared with SOTA methods. The results of CLIC dataset and Tecnick dataset are shown in Fig. 8 and Fig. 9, respectively. We also achieve similar good results on these two datasets. These results suggest that our method is robust and can achieve SOTA performance based on all of the three datasets with different resolutions. To get quantitative results, we present the BD-rate [4] computed from PSNR-BPP curves as the quantitative metric. The anchor RD-performance is set as the results of VVC on different datasets (BD-rate=0%). Our method outperforms VVC (VTM-12.1) by 12.30%, 13.71%, 11.85% in BD-rate on Kodak, Tecnick, and CLIC datasets, respectively. Table 1 shows partial results on Kodak. More comparisons are reported in Supplementary.

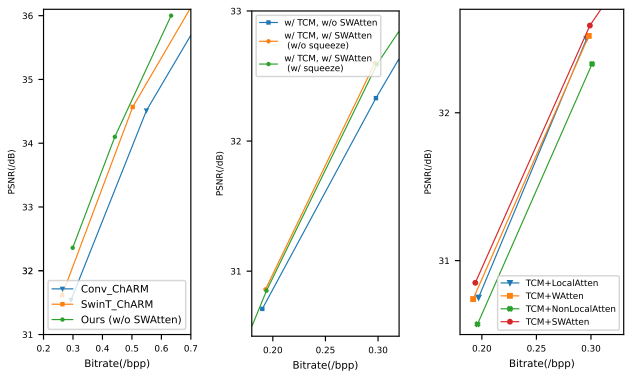
4.3. Ablation Studies

4.3.1 Comparison with Transformer-only/CNN-only based Models

In order to show the effectiveness of our proposed Transformer-CNN Mixture (TCM) blocks, we compare our medium model without SWAtten modules to the Transformer-only based model and CNN-only based model in [36]. The results are shown in Fig. 10a. "Conv_ChARM" and "SwinT_ChARM" are a CNN-only based model and a Transformer-only based model, respectively. They have similar architectures to our methods without SWAtten modules. The difference is that "SwinT_ChARM" uses Swin Transformer block, "Conv_ChARM" uses convolutional neural networks, and we use the proposed TCM block. By using the advantages both of transformer and CNN, the results show that our method surpasses the Transformer-only based model and CNN-only based model.

4.3.2 SWAtten Module

In Fig. 10b, we compare the cases using SWAtten modules or not. It can be observed that SWAtten modules bring a significant gain in RD-performance. Meanwhile, by using the channel squeeze operation in SWAtten modules, we can get a comparable performance compared with the situation without using that operation while saving many parameters. Section 4.5 shows more information on the parameters efficiency gain of this operation.



(a) (b) (c)

Figure 10. Experiments on Kodak dataset. (a) Comparison with Transformer-only/CNN-only based Models. (b) The ablation studies on SWAtten module ("w/o squeeze" represents that we don't add the channel squeeze/unsqueeze operation to SWAtten). (c) The RD-performance of models using different attention modules.

4.4. Various Attention Modules

In Fig. 10c, we compared our proposed SWAtten with the previous attention modules, including the non-local attention (NonlocalAtten) module [21], the local attention (LocalAtten) module [6], and the window-based attention (WAtten) module [37]. Compared with these different attention modules, SWAtten gets the best RD-performance because it can capture non-local information while also paying enough attention to local information.

4.5. Complexity and Qualitative Results

We test the complexity and qualitative results of different methods based on Kodak. Two other SOTA works [34, 36], are also tested as Table 1 shows. The results of our method suggest that the efficiency and RD-performance of our method can outperform both of these two methods. Meanwhile, after using channel squeeze in SWAtten, we can save a lot of parameters and FLOPs, while we can get a comparable BD-rate. It also should be noted that all of our small, medium and large models can achieve SOTA RD-performance. Meanwhile, the performance can further improve as the complexity increases, which shows that our model has a lot of potentials.

Table 1. The RD-performance and complexity of learned image compression models based on Kodak using GPU (RTX 3090). A lower BD-rate indicates higher RD-performance.

Methods	**Encoding Time(/ms)**	**Decoding Time(/ms)**	**Parameters(/M)**	**FLOPs (/G)**	**BD-rate**
Xie *et al*. [34]	2346	5212	47.55	408.21	-1.65
SwinT _ChARM [36]	132	84	60.55	230.37	-4.02
Ours (Large, w/o squeeze)	151	141	160.45	830.90	-12.54
Ours(Large)	150	140	75.89	700.96	-12.30
Ours(Medium)	130	122	58.72	415.20	-9.65
Ours(Small)	109	102	44.96	211.54	-7.39

4.6. Visualization

Fig. 1 shows the visualization example of decompressed images ($kodim23$) from Kodak dataset by our methods and the classical compression standards WebP, VVC (VTM 12.1) [29]. In some complex texture parts, our methods can keep more details (clearer feather outline).

5. Conclusion

In this paper, we incorporate transformers and CNN to propose an efficient parallel transformer-CNN mixture block that utilizes the local modeling ability of CNN and the non-local modeling ability of transformers. Then, a new image compression architecture is designed based on the TCM block. Besides, we present a swin-transformer-based attention module to improve channel-wise entropy models. The results of experiments show that the image compression model with TCM blocks outperforms the CNN-only/Transformer-only based models under a suitable complexity. Furthermore, the performance of SWAtten surpasses previous attention modules designed for image compression. At last, our method achieves state-of-the-art on three different resolution datasets (i.e., Kodak, Tecnick, CLIC Professional Validation) and is superior to existing image compression methods.

6. Acknowledgment

This paper is supported by Japan Science and Technology Agency (JST), under Grant JPMJPR19M5; Japan Society for the Promotion of Science (JSPS), under Grant 21K17770; Kenjiro Takayanagi Foundation; the Foundation of Ando Laboratory; NICT, Grant Number 03801, Japan.

References

[1] Nicola Asuni and Andrea Giachetti. Testimages: a large-scale archive for testing visual devices and basic image processing algorithms. In *STAG*, pages 63–70, 2014. 6

[2] Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. In *International Conference on Learning Representations*, 2016. 2

[3] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. In *Proceedings of the International Conference on Learning Representations*, 2018. 1, 2, 7, 11

[4] Gisle Bjontegaard. Calculation of average psnr differences between rd-curves. In *VCEG-M33*, 2001. 2, 7, 11

[5] Fangdong Chen, Yumeng Xu, and Li Wang. Two-stage octave residual network for end-to-end image compression. In *Proceedings of AAAI conference on artificial intelligence*, 2022. 1, 2, 7, 11

[6] Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. Learned image compression with discretized gaussian mixture likelihoods and attention modules. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7939–7948, 2020. 1, 2, 3, 4, 7, 8, 11, 12, 13, 14

[7] CLIC. Workshop and challenge on learned image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 6

[8] Jia Deng. A large-scale hierarchical image database. *Proceedings of IEEE/CVF conference on Computer Vision and Pattern Recognition*, 2009. 6

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020. 1, 2

[10] Haisheng Fu, Feng Liang, Jianping Lin, Bing Li, Mohammad Akbari, Jie Liang, Guohe Zhang, Dong Liu, Chengjie Tu, and Jingning Han. Learned image compression with discretized gaussian-laplacian-logistic mixture model and concatenated residual modules. *arXiv preprint arXiv:2107.06463*, 2021. 1

[11] Google. Web picture format. 2010. 1

[12] Zongyu Guo, Zhizheng Zhang, Runsen Feng, and Zhibo Chen. Causal contextual prediction for learned image compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(4):2329–2341, 2021. 2

[13] Dailan He, Ziming Yang, Weikun Peng, Rui Ma, Hongwei Qin, and Yan Wang. 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*, 2022. 1, 3, 5, 7, 11

[14] Dailan He, Yaoyan Zheng, Baocheng Sun, Yan Wang, and Hongwei Qin. Checkerboard context model for efficient learned image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14771–14780, 2021. 2, 4

[15] Jun-Hyuk Kim, Byeongho Heo, and Jong-Seok Lee. Joint global and local hierarchical priors for learned image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5992–6001, 2022. 2, 7, 11

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[17] Eastman Kodak. Kodak lossless true color image suite (photocd pcd0992). 1993. 6

[18] A Burakhan Koyuncu, Han Gao, and Eckehard Steinbach. Contextformer: A transformer with spatio-channel attention for context modeling in learned image compression. In *European Conference on Computer Vision*, 2022. 2

[19] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1833–1844, 2021. 2

[20] Fangzheng Lin, Heming Sun, Jinming Liu, and Jiro Katto. Multistage spatial context models for learned image compression. *arXiv preprint arXiv:2302.09263*, 2023. 1

[21] Haojie Liu, Tong Chen, Peiyao Guo, Qiu Shen, Xun Cao, Yao Wang, and Zhan Ma. Non-local attention optimized deep image compression. *arXiv preprint arXiv:1904.09757*, 2019. 1, 2, 8

[22] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 1, 2, 4

[23] Ming Lu, Peiyao Guo, Huiqing Shi, Chuntong Cao, and Zhan Ma. Transformer-based image compression. *arXiv preprint arXiv:2111.06707*, 2021. 1

[24] Ao Luo, Heming Sun, Jinming Liu, and Jiro Katto. Memory-efficient learned image compression with pruned hyperprior module. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 3061–3065, 2022. 6

[25] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks. *Advances in neural information processing systems*, 29, 2016. 4

[26] David Minnen, Johannes Ballé, and George D Toderici. Joint autoregressive and hierarchical priors for learned image compression. *Advances in neural information processing systems*, 31, 2018. 2, 3, 11, 12

[27] David Minnen and Saurabh Singh. Channel-wise autoregressive entropy models for learned image compression. In *IEEE International Conference on Image Processing (ICIP)*, pages 3339–3343. IEEE, 2020. 2, 3, 4, 5, 6, 11, 12

[28] Yichen Qian, Xiuyu Sun, Ming Lin, Zhiyu Tan, and Rong Jin. Entroformer: A transformer-based entropy model for learned image compression. In *International Conference on Learning Representations*, 2021. 2

[29] Joint Video Experts Team. Vvc official test model vtm. 2021. 1, 7, 8

[30] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021. 2

[31] Gregory K Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992. 1

[32] Dezhao Wang, Wenhan Yang, Yueyu Hu, and Jiaying Liu. Neural data-dependent transform for learned image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17379–17388, 2022. 1

[33] Yaojun Wu, Xin Li, Zhizheng Zhang, Xin Jin, and Zhibo Chen. Learned block-based hybrid image compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 2021. 1

[34] Yueqi Xie, Ka Leong Cheng, and Qifeng Chen. Enhanced invertible encoding for learned image compression. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 162–170, 2021. 1, 2, 6, 7, 8, 11

[35] Kai Zhang, Yawei Li, Jingyun Liang, Jiezhang Cao, Yulun Zhang, Hao Tang, Radu Timofte, and Luc Van Gool. Practical blind denoising via swin-conv-unet and data synthesis. *arXiv preprint arXiv:2203.13278*, 2022. 2

[36] Yinhao Zhu, Yang Yang, and Taco Cohen. Transformer-based transform coding. In *International Conference on Learning Representations*, 2022. 1, 2, 3, 7, 8, 11

[37] Renjie Zou, Chunfeng Song, and Zhaoxiang Zhang. The devil is in the details: Window-based attention for image compression. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022. 1, 2, 4, 7, 8, 11, 13, 14

A. Classical Image Compression Standard Setting

A.1. VVC

We use VTM-12.1 which is built form the website[4] to achieve VVC. The script from CompressAI [5] is utilized to evaluate model. The command is as the following:

```
python -m compressai.utils.bench vtm [
    path of image folder];
-c [path of VTM folder]/cfg/
    encoder_intra_vtm.cfg
-b [path of VTM folder]/bin
-q 16, 18, 20, 22, 24, 26, 28, 30, 32,
     34, 36, 38, 40
```

A.2. WebP

We use the API of Pillow (PIL) to achieve WebP algorithm. The code is:

```
img.save(REC_WEBP, 'webp', quality=
    quality)
```

where quality is set as $\{5,10,15,20,25,30,35,40,45,50\}$.

B. Detailed Network Architecture

The architecture of the our method is shown in Fig. 11. The head dimensions of TCM blocks in g_a and g_s are set as $\{8, 16, 32, 32, 16, 8\}$, while the head dimensions of TCM blocks in h_a and h_s are set as 32. We set channel numbers C of TCM blocks as 128/192/256 for Small/Medium/Large model. RBS and RBU have the same architectures as in [6]. The numbers of channels of the middle convolutional layers in RBS and RBU are 64/96/128 for our Small/Medium/Large model, while the number of last layer of RBS and RBU is 128/192/256. Here, to achieve the balance between running speed and RD-performance, we reduce the slices number in [27] from 10 to 5. Therefore, we have 5 Channel-Conditional Parameter Nets with SWAtten to get $\{\boldsymbol{\mu_0}, \boldsymbol{\mu_1}, \boldsymbol{\mu_2}, \boldsymbol{\mu_3}, \boldsymbol{\mu_4}\}$ and $\{\boldsymbol{\sigma_0}, \boldsymbol{\sigma_1}, \boldsymbol{\sigma_2}, \boldsymbol{\sigma_3}, \boldsymbol{\sigma_4}\}$. Also, we have 5 Latent Residual Prediction to get $\{\overline{\boldsymbol{y}}_0, \overline{\boldsymbol{y}}_1, \overline{\boldsymbol{y}}_2, \overline{\boldsymbol{y}}_3, \overline{\boldsymbol{y}}_4\}$. All the restored slices are concatenated as $\overline{\boldsymbol{y}}$ which is sent to decoder g_s to get a decompressed image.

C. Comparison with Recent LIC Works

To get quantitative results, we present the BD-rate [4] computed from PSNR-BPP curves as the quantitative metric. The anchor RD-performance is set as the results of VVC on different datasets (BD-rate=0%). The Table 2 shows the results. As results show, we outperform the previous works and achieve SOTA performance based on the three datasets with different resolutions.

[4]https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/-/releases/VTM-12.1

[5]https://github.com/InterDigitalInc/CompressAI/tree/master/compressai/utils/bench

Table 2. BD-rate improvements against the VVC anchor. For different datasets, the anchor is recalculated based the corresponding dataset. Lower BD-rate represents higher performance.

Methods	Dataset	BD-Rate
Cheng *et al.* [6]	Kodak 768x512	3.16
Xie *et al.* [34]		-1.65
Chen *et al.* [5]		-6.21
He *et al.* [13]		-7.49
Ours (Large)		**-12.30**
Ours (Medium)		**-9.65**
Ours (Small)		**-7.39**
Ballé *et al.* [3]	Tecnick 1200x1200	30.66
Xie *et al.* [34]		-4.07
Kim *et al.* [15]		6.98
Ours (Large)		**-13.71**
Ours (Medium)		**-11.29**
Ours (Small)		**-9.53**
Cheng *et al.* [6]	CLIC-P val 2K	6.77
Xie *et al.* [34]		-2.60
Chen *et al.* [5]		-7.15
Zou *et al.* [37]		-3.68
Ours (Large)		**-11.85**
Ours (Medium)		**-10.27**
Ours (Small)		**-8.94**
VVC	-	0

D. Ablation Studies on Various Entropy Estimation Models

To verify our TCM blocks can improve the overall RD-performance, in addition to test the model with the channel-wise entropy model in [27], we also try the model using the spatial-wise entropy model in [26]. The results are shown in Fig. 12. We define the model where the main path uses TCM block as "TCMmain". We compare the TCM-main model using spatial-wise entropy model with "SwinT-Hyperprior" model in [36] and the model in [6]. All of these three methods use spatial-wise entropy models. The difference is that our model is based on TCM block, the model in [6] is based on CNN, and "SwinT-Hyperprior" is based on swin-transformer. As we can see, our method can get the best RD-performance. This suggests that the TCM blocks can significantly improve image compression, and are robust to different entropy models.

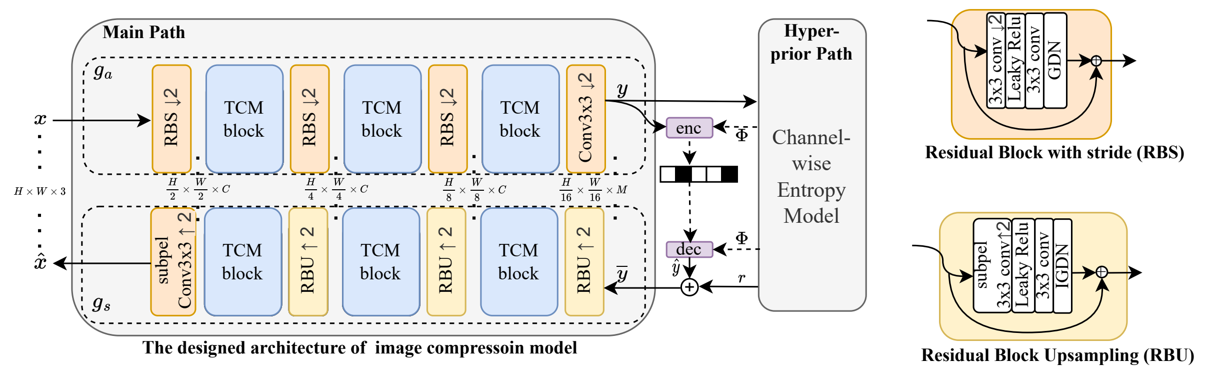

Main Path
g_a
x
RBS ↓2
TCM block
RBS ↓2
TCM block
RBS ↓2
TCM block
Conv3x3 ↓2
y
$\frac{H}{2}\times\frac{W}{2}\times C$
$\frac{H}{4}\times\frac{W}{4}\times C$
$\frac{H}{8}\times\frac{W}{8}\times C$
$\frac{H}{16}\times\frac{W}{16}\times M$
$H\times W\times 3$
$\hat{x}$
subpel Conv3x3 ↑2
TCM block
RBU ↑2
TCM block
RBU ↑2
TCM block
RBU ↑2
$\bar{y}$
g_s
enc
dec
Φ
$\hat{y}$
r
Hyper-prior Path
Channel-wise Entropy Model
The designed architecture of image compressoin model
3x3 conv ↓2
Leaky Relu
3x3 conv
GDN
Residual Block with stride (RBS)
subpel 3x3 conv↑2
Leaky Relu
3x3 conv
IGDN
Residual Block Upsampling (RBU)

Figure 11. The overall framework (left). The architectures of RBS and RBU in [6] (right).

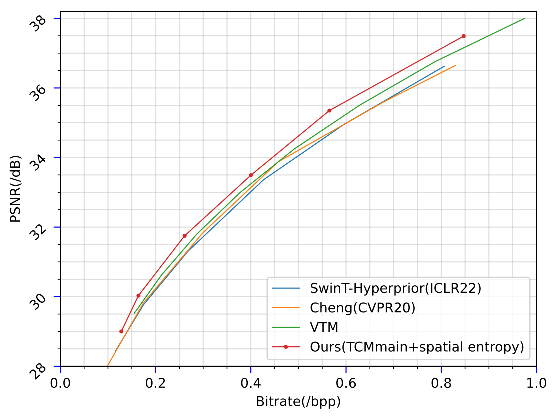


Figure 12. Performance evaluation of the models using the spatial-wise entropy model [26] on the Kodak dataset.

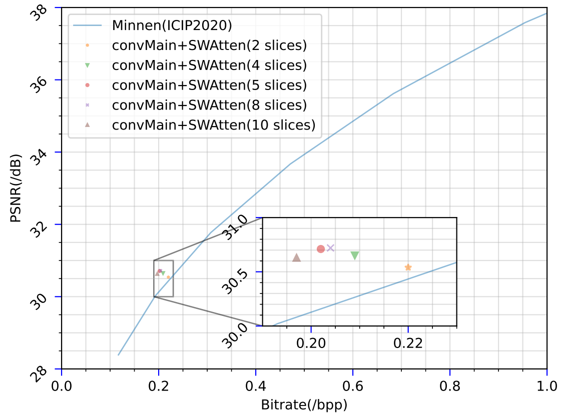


Figure 13. Effect of different slices numbers on RD-performance. “convMain” means we use the main path in [27].

E. Ablation Studies on the Numbers of Slices

The number of slices s is an important hyper-parameter for channel-wise entropy model in [27]. A larger number leads to lower efficiency, while a lower number causes a worse RD-performance. To find a suitable number, we test some different number setting $s = \{2, 4, 5, 8, 10\}$ for our entropy model with the proposed SWAtten. The main path of the tested model is the same as the main path in [27]. The entropy model is also similar, the difference is that we add SWAtten. The results are shown in Fig. 13. As we can see, when s is low, we get a bad RD-performance. With s increasing, the performance is improved. But when $s > 5$, the improvement of the performance is not obvious, and even decreases. This indicates that 5 slices have been able to learn enough information in our model. Therefore, we set s as 5 in our model to achieve the balance between running speed and RD-performance.

F. Abaltion Studies on the Design of SWAtten

We test the SWAtten w/o CNN for attention map (green point) in Fig. 14. In addition, we also evaluate the case w/o the swin transformer (yellow point). From the results, we can see that using either of these two modules can bring about 0.1dB PSNR improvement with fewer bitrates.

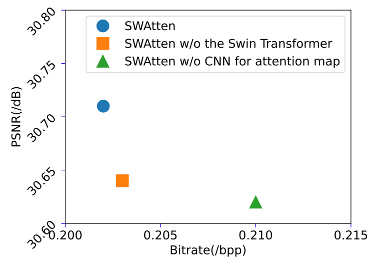


Figure 14. The ablation study on SWAtten (using the same main path as [27].

G. Visualization

We conducted a comparison between our TCM-based model and both a CNN-based [6] and Transformer-based [37] model using the Kodak dataset's $kodim19$ and $kodim20$ images. The results of this comparison are presented in Fig. 15. We focused our analysis on two local regions, and the differences between the three models are noticeable. In the upper local area of $kodim19$, our method effectively reconstructed some of the road sign details while making it easier to differentiate between the back fences. The pasted poster color is also clearer and not distorted. In contrast, our method generated fewer artifacts compared to the CNN-based/Transformer-based models when reconstructing the lower local region. For $kodim20$, our method generated the clearest sign in the left local region, and we achieved a clearer "E" letter than the other two methods in the right local region.

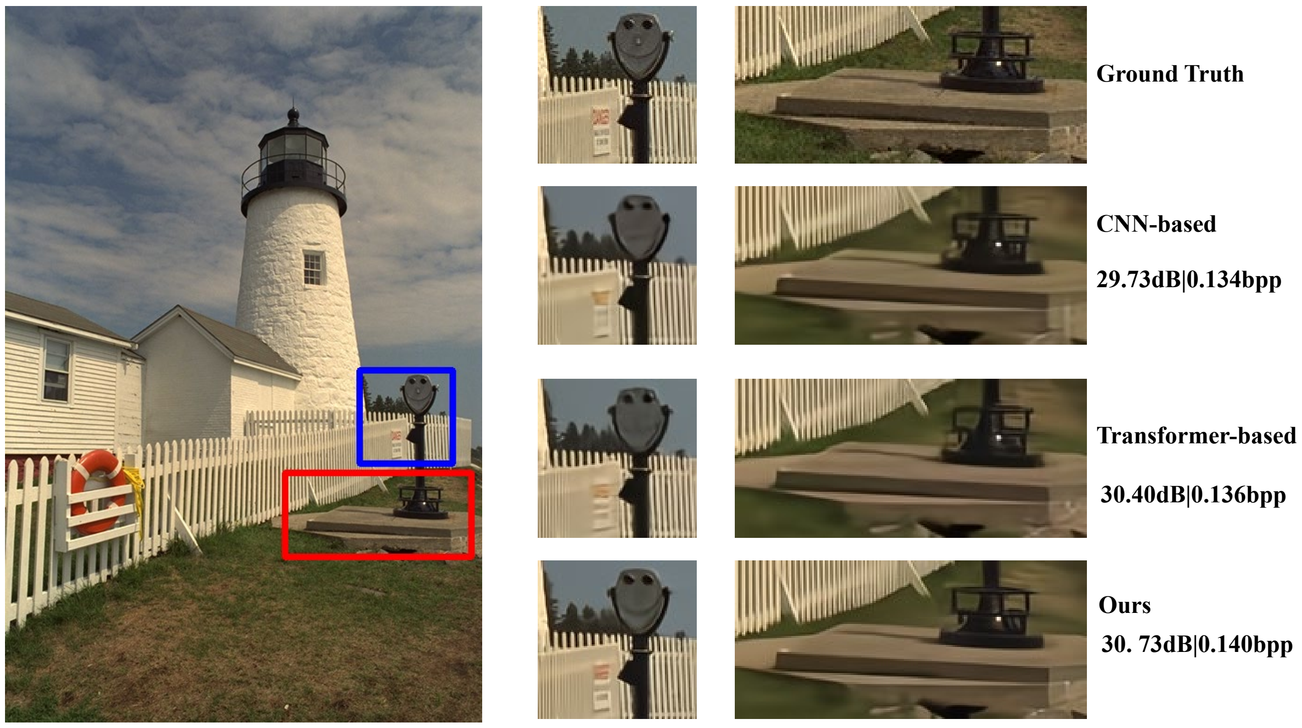


Figure 15. The visualization of $kodim19$ in Kodak dataset by using our TCM-based model, CNN-based model [6], and transformer-based model [37]. PSNR|Bit-rate is listed in the last column.

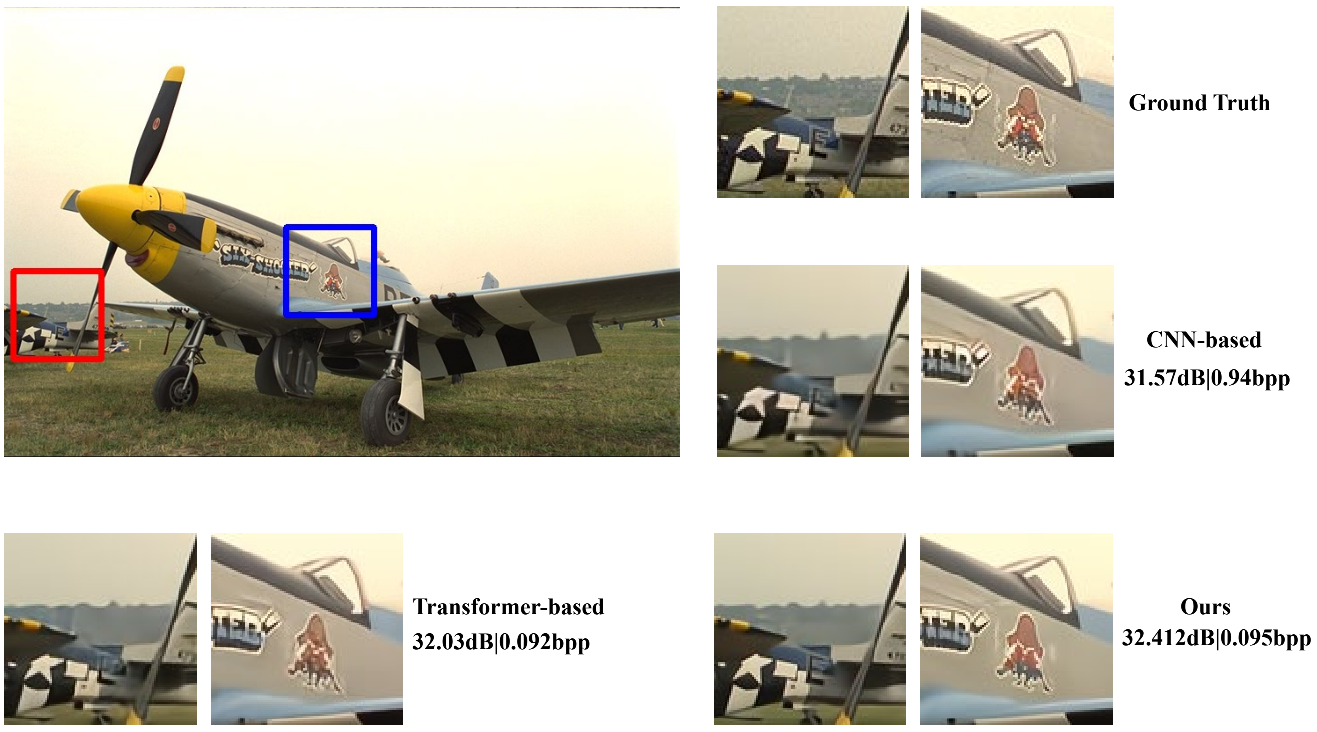


Figure 16. The visualization of $kodim20$ in Kodak dataset by using our TCM-based model, CNN-based model [6], and transformer-based model [37]. PSNR|Bit-rate is listed on the subfigures' right.