# Checkerboard Context Model for Efficient Learned Image Compression

Dailan He    Yaoyan Zheng    Baocheng Sun    Yan Wang*    Hongwei Qin
SenseTime Research
{hedailan, zhengyaoyan, sunbaocheng, wangyan1, qinhongwei}@sensetime.com

## Abstract

*For learned image compression, the autoregressive context model is proved effective in improving the rate-distortion (RD) performance. Because it helps remove spatial redundancies among latent representations. However, the decoding process must be done in a strict scan order, which breaks the parallelization. We propose a parallelizable checkerboard context model (CCM) to solve the problem. Our two-pass checkerboard context calculation eliminates such limitations on spatial locations by re-organizing the decoding order. Speeding up the decoding process more than 40 times in our experiments, it achieves significantly improved computational efficiency with almost the same rate-distortion performance. To the best of our knowledge, this is the first exploration on parallelization-friendly spatial context model for learned image compression.*

## 1. Introduction

Image compression is a vital and long-standing research topic in multimedia signal processing. Various algorithms are designed to reduce spatial, visual, and statistical redundancies to produce more compact image representations. Common image compression algorithms like JPEG [15], JPEG2000 [30] and BPG [8] all follow a general pipeline, where lossless entropy coders [23] are used after image transformations and quantization. In those non-learned image compression methods, content loss only occurs in the quantization process. The transformations involved mainly include Discrete Cosine Transformation and Wavelet Transformation, which are lossless.

In recent years, many state-of-the-art (SOTA) deep learning and computer vision techniques have been introduced to build powerful learned image compression methods. Many studies aim to establish novel image compression pipelines based on recurrent neural networks [35], convolutional autoencoders [5, 34, 31, 3, 6], or generative adversarial networks [1]. Some of them [26, 18, 11] have attained a better
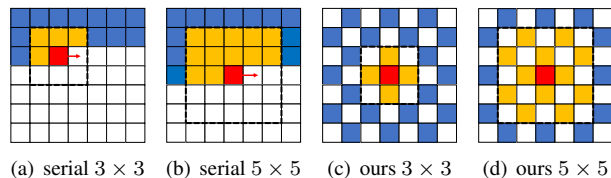
Figure 1. Masked convolutions modeling spatial causal context. The red blocks denote elements to en/de-code. Latents in yellow and blue locations are currently visible (all of them are visible during encoding, and those who have been decoded are visible during decoding). A context modeling can be conducted using a masked convolution which is centered at the red location and only convolves with the yellow latents. (a)(b) $3 \times 3$ and $5 \times 5$ instances of the widely used serial context model. This context model requires strict Z-ordered serial decoding, which limits the computational efficiency. (c)(d) Our proposed checkerboard context model with a kernel size of $3 \times 3$ and $5 \times 5$. After decoding all anchors, which are latents in blue and yellow locations, the context calculating for all non-anchors can be run in parallel.

performance than those currently SOTA conventional compression techniques such as JPEG2000 [30] and BPG [8] on both the peak signal-to-noise ratio (PSNR) and multi-scale structural similarity (MS-SSIM) [36] distortion metrics. Of particular note is, even the intra coding of Versatile Video Coding (VVC) [9], an upcoming video coding standard, has been approached by a recent learning-based method [10].

The common key of those currently most successful approaches is the entropy modeling and optimizing method, with autoencoder based structure to perform a nonlinear transform coding [14, 5]. By estimating the probability distribution of latent representations, such models can minimize the entropy of these representations to be compressed, which directly correlates to the final code length using arithmetic encoding [32] or range encoding [24], and enable a differentiable form of rate-distortion (RD) optimization. Another important aspect is the introduction of hyperprior [6]. Hyper latent is the further extracted representation, which provides side-information implicitly describing the spatial correlations in latent. Adopting hyperprior modeling allows entropy models to approximate latent distributions more precisely and benefits the overall coding perfor-

mance. This method [6] is referred to as a scale hyperprior framework in their later work [26], where hyper latent is used to predict the entropy model's scale parameter.

The context model [26, 18], inspired by the concept of *context* from traditional codecs, is used to predict the probability of unknown codes based on latents that have already been decoded (as shown in Figure 1(a-b)). This method is referred to as a mean-scale hyperprior framework [26, 16], where hyper latent and context are used jointly to predict both the location (*i.e.* mean value) and scale parameter of the entropy model. Evaluated by previous works [26, 18], combining all the above-mentioned components (differentiable entropy modeling, hyper latent, and context model) can beat BPG in terms of PSNR and MS-SSIM. These context models are extended to more powerful and, of course, more computation-consuming ones by a series of later works [38, 19, 11, 10].

Though it seems promising, there are still many problems to solve before those models can be used in practice. The above-mentioned context model, which plays a crucial role in achieving SOTA performance and is adopted by most recent works, has a horribly low computational efficiency because of the lack of parallelization [26, 16]. Recent works focusing on the real-world deployment of practical neural image compression choose to omit the context model due to its inefficiency. They choose to use only the scale hyperprior framework [4] or the mean-scale hyperprior framework but without the context model [16]. Li *et al.* introduce CCN [21] for a faster context calculation with moderate parallelizability but its efficiency is still limited by image size. In order to develop a practical and more effective learning-based image codec, it is essential to investigate more efficient context models.

In this paper, we propose a novel parallelizable checkerboard context model along with a two-pass decoding method to achieve a better balance between RD performance and running efficiency.

## 2. Related Work

The most related works, including [5, 6, 26], which establish a powerful convolutional autoencoder framework for learned image compression, have been introduced in the introduction part.

A very recent work [27] proposes a channel-wise autoregressive entropy model that aims to minimize the element-level serial processing in the context model of [26] and achieves SOTA RD performance when combined with latent residual prediction and round-based training. Our checkerboard context is built in the spatial dimension, which provides a new solution to the serial processing problem.

There are also related works aiming to improve different aspects of the convolutional autoencoder framework, in-
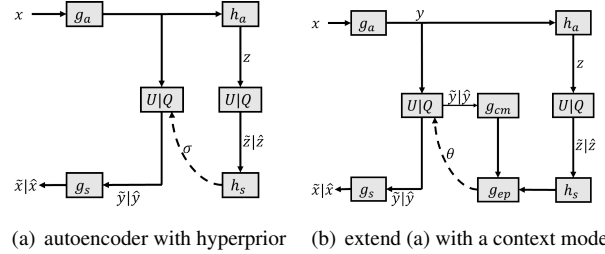


(a) autoencoder with hyperprior   (b) extend (a) with a context model

Figure 2. Operational diagrams: (a) Autoencoder with hyperprior. Arrows show the data flow direction. Boxes are transforms implemented by neural networks. Boxes labeled $U|Q$ is the quantization module, which performs uniform perturbation in the training stage and quantization in the inference stage. (b) Autoencoder with both hyperprior and context model.

cluding using more complex entropy models [25, 28, 20, 10], enhancing reconstruction quality with post-processing networks [39], enabling variable rate compression using a single model [11, 13], using content aware compression [22], and incorporating multi-scale or attention-based neural architectures and fancier quantization methods [25, 40, 37].

## 3. Preliminary

### 3.1. Variational Image Compression with Hyperprior

The diagram for the scale hyperprior framework [6] is given in Figure 2(a). $g_a, g_s, h_a, h_s$ are nonlinear transforms implemented by neural networks. $\boldsymbol{x}$ is the original image, $\boldsymbol{y} = g_a(\boldsymbol{x})$ and $\hat{\boldsymbol{y}} = Q(\boldsymbol{y})$ are latent representations before and after quantization, $\hat{\boldsymbol{x}} = g_s(\hat{\boldsymbol{y}})$ is the reconstructed image. $\boldsymbol{z} = h_a(\boldsymbol{y})$ and $\hat{\boldsymbol{z}} = Q(\boldsymbol{z})$ are the hyper latent before and after quantization. $\hat{\boldsymbol{z}}$ is used as side information to estimate the scale parameter $\boldsymbol{\sigma}$ for the entropy model of latent $\hat{\boldsymbol{y}}$. During training, the quantization operation is approximated by adding uniform noise, producing differentiable variables $\tilde{\boldsymbol{y}}$, $\tilde{\boldsymbol{z}}$ and $\tilde{\boldsymbol{x}}$. Hereinafter we always use $\hat{\boldsymbol{x}}$, $\hat{\boldsymbol{y}}$ and $\hat{\boldsymbol{z}}$ to represent $\tilde{\boldsymbol{x}}|\hat{\boldsymbol{x}}$, $\tilde{\boldsymbol{y}}|\hat{\boldsymbol{y}}$ and $\tilde{\boldsymbol{z}}|\hat{\boldsymbol{z}}$ for simplicity. The trade-off between rate and distortion or the loss function can be written as:

$$R + \lambda \cdot D = \mathbb{E}_{\boldsymbol{x} \sim p_{\boldsymbol{x}}}[-\log_2 p_{\hat{\boldsymbol{y}}|\hat{\boldsymbol{z}}}(\hat{\boldsymbol{y}}|\hat{\boldsymbol{z}}) - \log_2 p_{\hat{\boldsymbol{z}}}(\hat{\boldsymbol{z}})] \\ + \lambda \cdot \mathbb{E}_{\boldsymbol{x} \sim p_{\boldsymbol{x}}}[d(\boldsymbol{x}, \hat{\boldsymbol{x}})] \quad (1)$$

where bit rate of latent $\hat{\boldsymbol{y}}$ and hyper latent $\hat{\boldsymbol{z}}$ is approximated by estimated entropy, $\lambda$ controls the bit rate(*i.e.* larger $\lambda$ for larger rate and better reconstruction quality), $d(\boldsymbol{x}, \hat{\boldsymbol{x}})$ is the distortion term, usually using MSE or MS-SSIM.

With scale hyperprior [6], the probability of latents $\hat{\boldsymbol{y}}$ can be modeled by a conditional Gaussian scale mixture (GSM) model:

$$p_{\hat{\boldsymbol{y}}|\hat{\boldsymbol{z}}}(\hat{y}_i|\hat{\boldsymbol{z}}) = \left[ \mathcal{N}(\mu_i, \sigma_i^2) * \mathcal{U}(-\frac{1}{2}, \frac{1}{2}) \right](\hat{y}_i) \quad (2)$$

$$p_{\hat{\boldsymbol{y}}|\hat{\boldsymbol{z}}}(\hat{\boldsymbol{y}}|\hat{\boldsymbol{z}}) = \prod_i p_{\hat{\boldsymbol{y}}|\hat{\boldsymbol{z}}}(\hat{y}_i|\hat{\boldsymbol{z}}) \qquad (3)$$

where the location parameter $\mu_i$ is assumed to be zero and the scale parameter $\sigma_i$ is the i-th element of $\boldsymbol{\sigma} = h_s(\hat{\boldsymbol{z}})$ for each code in $\hat{\boldsymbol{y}}$ given the hyperprior. The probability of the hyper latent $\hat{\boldsymbol{z}}$ can be modeled using a non-parametric fully factorized density model [6].

## 3.2. Autoregressive Context

In the mean-scale hyperprior framework [26], an additional module called context model is added to boost the RD performance. Figure 2(b) shows the entire structure consisting of autoencoder $(g_a, g_s)$, hyper autoencoder $(h_a, h_s)$, context model $(g_{cm})$, and a parameter inference network $(g_{ep})$ which estimates the location and scale parameters $\boldsymbol{\Phi} = (\boldsymbol{\mu}, \boldsymbol{\sigma})$ of the entropy model for latent $\hat{\boldsymbol{y}}$. Let $h_s(\hat{\boldsymbol{z}})$ denote the hyperprior feature and $g_{cm}(\hat{\boldsymbol{y}}_{<i})$ denote the context feature, the parameter prediction for $i$-th representation $\hat{y}_i$ is

$$\boldsymbol{\Phi}_i = (\mu_i, \sigma_i) = g_{ep}\big(h_s(\hat{\boldsymbol{z}}), g_{cm}(\hat{\boldsymbol{y}}_{<i})\big) \qquad (4)$$

where $\hat{\boldsymbol{y}}_{<i}$ means the causal context (*i.e.* some nearby visible latents of latent $\hat{y}_i$). This type of context model can be implemented using masked convolutions. Given the $k \times k$ binary mask $\boldsymbol{M}$ and convolutional weights $\boldsymbol{W}$, the masked convolution with kernel size of $k \times k$ on input $\boldsymbol{x}$ can be performed in a reparameterization form:

$$g_{cm}(\boldsymbol{x}) = (\boldsymbol{M} \odot \boldsymbol{W})\boldsymbol{x} + \boldsymbol{b} \qquad (5)$$

where $\odot$ is the Hadamard operator and $\boldsymbol{b}$ is a bias term. Since $\boldsymbol{M}$ describes the context modeling pattern, by manually set different mask $\boldsymbol{M}$ various context referring schemes are obtained. In previous works, usually the mask shown in Figure 1(a) is adopted as $\boldsymbol{M}$ to implement a left-top reference which requires strict Z-ordered serial decoding because only already decoded latents are visible.

In [10], the entropy model is improved by using a discretized K-component Gaussian mixture model (GMM):

$$p_{\hat{\boldsymbol{y}}|\hat{\boldsymbol{z}}}(\hat{y}_i|\hat{\boldsymbol{z}}) = \sum_{0<k<K} \pi_i^{(k)} \left[ \mathcal{N}(\mu_i^{(k)}, \sigma_i^{2(k)}) * \mathcal{U}(-\frac{1}{2}, \frac{1}{2}) \right] (\hat{y}_i)$$
$$(6)$$

where K groups of entropy parameters $(\boldsymbol{\pi}^{(k)}, \boldsymbol{\mu}^{(k)}, \boldsymbol{\sigma}^{(k)})$ are calculated by $g_{ep}$. Combined with the autogressive context model, [10] is the first to achieve comparable PSNR with VVC.

Following previous works [26, 10], we do not apply context model to the hyper latent $\hat{\boldsymbol{z}}$.

## 4. Parallel Context Modeling

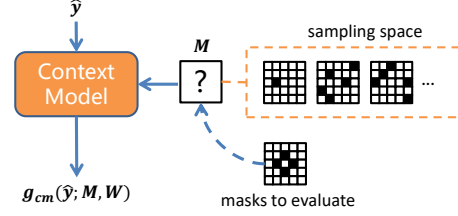Though the previous context model, called **serial** context model by us and shown in Figure 1(a), has a limita-



Figure 3. Context model in proposed random-mask model. Mask $\boldsymbol{M}$ is randomly generated during training (orange dashed box) and replaced by a fixed mask (blue dashed line) during evaluation.

tion on computational efficiency, most of the SOTA methods still rely on it [26, 10, 20]. Therefore, improvement is highly required. We firstly analyze the context model using a random-mask model, then we propose a novel parallel context model with checkerboard shaped masked convolution as a replacement to the existing serial context model.

### 4.1. Random-Mask Model: Test Arbitrary Masks

The context model can be seen as a convolution with weight $\boldsymbol{W}$ conditioned on the binary mask $\boldsymbol{M}$, as described in eq. 5 where the mask describes the pattern of context modeling. To understand the mechanism of context models and explore better modeling patterns, we propose a random-mask toy model to which arbitrary mask can be fed as the context modeling pattern.

The random-mask model is adapted from the above-mentioned autoregressive model [26] with the same en/decoders and $g_{ep}$. The serial context model with a manually designed mask is replaced by a convolution conditioned on randomly generated masks during training. As shown in Figure 3, we generate random sampled $5 \times 5$ masks $\boldsymbol{M}$ and compute Hadamard product of $\boldsymbol{M}$ and non-masked convolution weights $\boldsymbol{W}$ to obtain the masked convolution weights $\boldsymbol{M} \odot \boldsymbol{W}$ in every iteration of the training stage. After each time of backward propagation, the weights $\boldsymbol{W}$ will be updated guided by a random mask $\boldsymbol{M}$, which implicitly establishes a supernet consisting of all context models conditioned on $5 \times 5$ masks. Therefore, weights are shared among context models using different masks so that the trained random-mask model can be used to evaluate arbitrary masks during inference. After training, to test the performance of a particular mask pattern, we simply feed that mask to the context model as $\boldsymbol{M}$, and then the random-mask model has a context model with a fixed mask.

This random-mask model is suitable to measure the ability to save bit rate brought by various context modeling patterns. The same latent $\hat{\boldsymbol{y}}$ and reconstructed image $\hat{\boldsymbol{x}}$ are shared (because of the weight sharing) so that the same distortion is shared among context modeling patterns, enabling the direct comparison of bit rates. Let $R_0$ denote the non-reference bits per pixel (BPP) when context modeling is disabled by inputting a full-zero mask to the random-mask

model (corresponds to the context-free mean-scale hyperprior baseline [26]). We further quantitate the ability of rate saving as a rate saving ratio:

$$\eta(\boldsymbol{M}) = \frac{R_0 - R_{\boldsymbol{M}}}{R_0} \times 100\% \qquad (7)$$

where $R_{\boldsymbol{M}}$ denotes the BPP feeding mask $\boldsymbol{M}$ to the random-mask model. In the following section, we use the trained random-mask model to analyze context modeling patterns represented by masks $\boldsymbol{M}$ by calculating their rate saving ratio $\eta(\boldsymbol{M})$.

## 4.2. How Distance Influences Rate Saving

The serial context model saves rate by referring to already decoded neighboring latents to estimate the entropy of current decoding latent more precisely. For such a model with a $5 \times 5$ mask (Figure 1(b)), latents on 12 locations at the left-top of the central location are referred to in the estimation. We find that latents on nearer locations contribute to rate saving much more significantly by calculating the $\eta$ of 24 different single-reference masks (each mask has only one location set to 1 and others set to 0, as shown in Figure 4(b)) based on the above-mentioned random-mask model. The result is shown in Figure 4(a). It is obvious that closer neighboring latents reduce much more bit rate, and neighbors with a distance of more than 2 elements influence the bit saving negligibly.

To dig deeper, we analyze how the context models help save rate. With context modeling, the entropy model estimates decoding latents $\hat{\boldsymbol{y}}_i$ conditioned on those visible already decoded latents $\hat{\boldsymbol{y}}_{j<i}$. Viewing those latents on different locations as samples from correlated virtual coding sources $\hat{Y}_i$ and $\hat{Y}_{j<i}$, the mutual information $I(\hat{Y}_i; \hat{Y}_{j<i})$ between them is recognized by context model and partially removed from bits used to encode $\hat{\boldsymbol{y}}_i$. According to the Slepian-Wolf coding theory [33], the optimal bit rate:

$$\hat{R} = H(\hat{Y}_i, \hat{Y}_{j<i}) = H(\hat{Y}_i) + H(\hat{Y}_{j<i}) - I(\hat{Y}_i; \hat{Y}_{j<i})$$

is theoretically reachable. Therefore, provided that the training is sufficient, a context modeling pattern referring to latents with more mutual information $I(\hat{Y}_i; \hat{Y}_{j<i})$, or simply saying, more causal relationship, saves more bit rate.

On the other hand, spatial redundancy is an important basis of image and video compression. Strong self-correlation exists in digital images, and adjacent pixels are likely to have a stronger causal relationship. Empirically, convolution outputs partially keep such redundancy because of locality [6], even when using scale hyperprior [26, 10]. So the latents still retain a similar redundancy.

That tells how the distance between referred latents and decoding latents influences the bit saving of context modeling in learned image compression. Nearer latents have
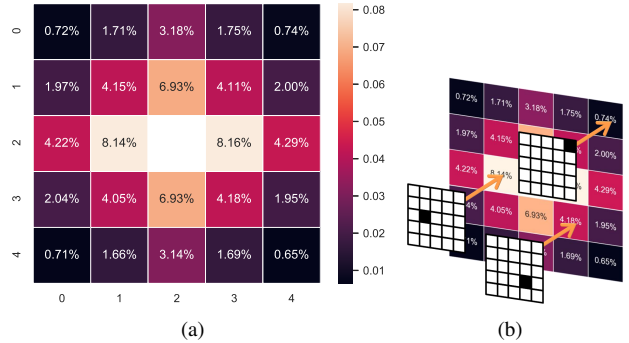


Figure 4. (a) Rate saving ratios of single-reference masks tested on Kodak images [17]. (b) For each single-reference mask, only one location is set to 1 (black) and others are set to 0 (white).

a stronger causal relationship, and more mutual information can be re-calculated during decoding instead of being stored and occupying the bit rate. Also, it helps explain why modeling the context using masked convolution with larger kernels or a stack of $5 \times 5$ convolutions (with a much larger receptive field) unexpectedly does damage to the RD-performance, as reported in previous work [26]. Referring to further neighbors that carry nearly no mutual information helps little but increases the risk of overfitting.

This implies that a context modeling pattern referring to more close neighbors is more likely to save more bit rate than presently adopted serial context models. We then establish our parallel context model based on this motivation.

## 4.3. Parallel Decoding with Checkerboard Context

Figure 1(c) shows a checkerboard shaped context model. Referring to four nearest neighbours, it outperforms both $3 \times 3$ and $5 \times 5$ serial context model in our further experiments on random-mask model (we will discuss this experiment in section 5.1 and Table 1, here we bring forward its conclusion to motivate the proposal of checkerboard shaped context models). Then we further extend it to a general form with arbitrary kernel size (for an instance of $5 \times 5$ kernel, see Figure 1(d)). Though it is impossible to apply it on the whole latent feature map (or the adjacent latents will depend on each other during decoding), it is helpful to building a computationally efficient parallel context model without introducing apparent RD performance loss compared with the present serial context model.

To develop our parallel decoding approach, we only encode/decode half of the latents (white and red ones in Figure 1(c) and Figure 1(d)) using checkerboard shaped context and hyperprior. The coding of the other half of latents, which we call **anchors**, only depends on the hyperprior. To implement these two sets of rules, we set the context feature of all anchors zero and adapt the calculation of entropy
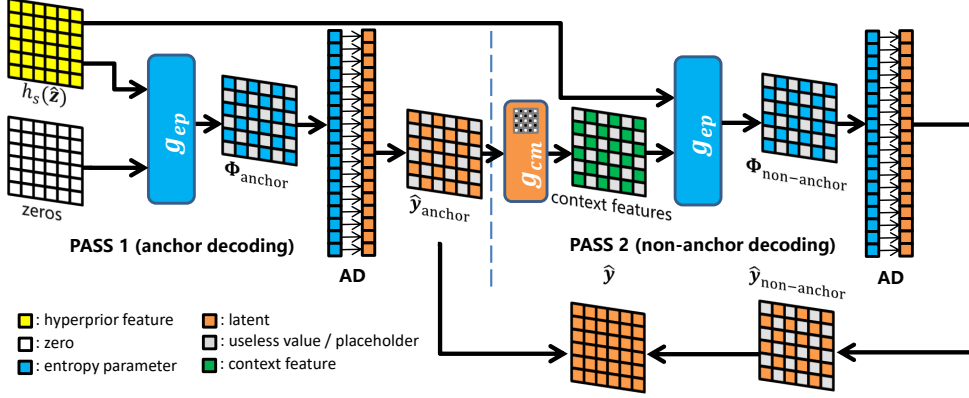
Figure 5. Illustration of the proposed two-pass decoding. $g_{cm}$ is the context model with a checkerboard mask and $g_{ep}$ is the parameter network. Entropy parameters $\mathbf{\Phi}_{\text{anchor}}$ and $\mathbf{\Phi}_{\text{non-anchor}}$ are estimated in turn. To decode the latents (orange blocks) from bitstream (not given in the diagram), useless values (gray blocks) in estimated entropy parameters are removed and the remained ones (blue blocks) are flattened and input into arithmetic decoder (AD). For further implementation details, please refer to the supplementary material.

parameters $\mathbf{\Phi}$ in eq. 4 to a spatial location conditioned form:

$$\mathbf{\Phi}_i = \begin{cases} g_{ep}\left(h_s(\hat{\boldsymbol{z}}), \mathbf{0}\right)_i, & \hat{y}_i \in \hat{\boldsymbol{y}}_{\text{anchor}} \\ g_{ep}\left(h_s(\hat{\boldsymbol{z}}), g_{cm}(\hat{\boldsymbol{y}}_{\text{anchor}}; \boldsymbol{M} \odot \boldsymbol{W})\right)_i, & \text{otherwise} \end{cases} \tag{8}$$

where $g_{cm}$ is the masked convolution as described in eq. 5 conditioned on a checkerboard-shaped mask $\boldsymbol{M}$. Its input $\hat{\boldsymbol{y}}_{\text{anchor}}$ is the set of anchors and $i$ is the index for the $i$-th element $\hat{y}_i$ in latent $\hat{\boldsymbol{y}}$. For approaches using mean-scale Gaussian entropy models, the entropy parameter $\Phi = (\boldsymbol{\mu}, \boldsymbol{\sigma})$, and for methods adopting GMM $\Phi$ consists of K groups of $\boldsymbol{\pi}^{(k)}$, $\boldsymbol{\mu}^{(k)}$ and $\boldsymbol{\sigma}^{(k)}$.

When anchors are visible, the context features of all non-anchors can be calculated in parallel by a masked convolution. Anchors' decoding is also run in parallel, so the entropy parameter calculation in eq. 4 for decoding can be performed in two passes, which is much more efficient than the serial context model.

### 4.3.1 Encoding Latents in One Pass

Here we reformulate eq. 8 to illustrate that the encoding process can be done within one pass. By one pass we mean entropy parameters $\mathbf{\Phi}$ of all the latents are obtained in parallel without element-wise sequential calculation. Let $\hat{\boldsymbol{y}}_{\text{half}}$ denotes the latents with all non-anchors set to zero:

$$(\hat{\boldsymbol{y}}_{\text{half}})_i = \begin{cases} \hat{y}_i, & \hat{y}_i \in \hat{\boldsymbol{y}}_{\text{anchor}} \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

Since now the binary mask $\boldsymbol{M}$ is checkerboard-shaped, if input $\hat{\boldsymbol{y}}_{\text{half}}$ into $g_{cm}$, according to eq. 5 we have:

$$g_{cm}(\hat{\boldsymbol{y}}_{\text{half}})_i = \begin{cases} b_{\hat{y}_i}, & \hat{y}_i \in \hat{\boldsymbol{y}}_{\text{anchor}} \\ g_{cm}(\hat{\boldsymbol{y}}_{\text{anchor}})_i, & \text{otherwise} \end{cases} \tag{10}$$

where $b_{\hat{y}_i} \in \boldsymbol{b}$ is the corresponding bias term added to $\hat{y}_i$. Similar to $\hat{\boldsymbol{y}}_{\text{half}}$, let $\boldsymbol{b}_{\text{half}}$ denote a feature map with all elements on anchor locations set to $\boldsymbol{b}$ and the others set to zeros. Because $g_{ep}$ is usually implemented as a point-wise transform consisting of a stack of $1 \times 1$ convolutions [26, 10], we can re-write eq. 8 to:

$$\mathbf{\Phi} = g_{ep}\left(h_s(\hat{\boldsymbol{z}}), g_{cm}(\hat{\boldsymbol{y}}_{\text{half}}) - \boldsymbol{b}_{\text{half}}\right) \tag{11}$$

During encoding (and training) all latents in $\hat{\boldsymbol{y}}$ are visible, so we can simply generate $\hat{\boldsymbol{y}}_{\text{half}}$ from $\hat{\boldsymbol{y}}$ by setting its non-anchors to zero and then calculate all entropy parameters $\mathbf{\Phi}$ in parallel with only one pass of context model and parameter network. Finally, we flatten and rearrange $\hat{\boldsymbol{y}}$ and $\mathbf{\Phi}$ and then encode anchors and non-anchors into the bitstream in turn. We will discuss cheap and parallel ways to get $\hat{\boldsymbol{y}}_{\text{half}}$ and $\boldsymbol{b}_{\text{half}}$ in the supplementary material. Hence, only an element-wise subtraction in eq. 11 is newly introduced to the encoding process for using the checkerboard context model, which won't slow down the encoding.

### 4.3.2 Decoding Latents in Two Passes

At the beginning of decoding, the hyper latent $\hat{\boldsymbol{z}}$ can be decompressed directly by an arithmetic decoder (AD) from the bitstream using code probability $p_{\hat{\boldsymbol{z}}}$. Then the hyperprior feature $h_s(\hat{\boldsymbol{z}})$ is calculated. After that, there are two decoding passes to obtain whole latent $\hat{\boldsymbol{y}}$, as shown in Figure 5.

In the first decoding pass, entropy parameters of anchors $\mathbf{\Phi}_{\text{anchor}}$ are calculated with the context feature set to zero according to eq. 8. Then the conditional probability $p_{\hat{\boldsymbol{y}}|\hat{\boldsymbol{z}}}$ of anchors is determined by these entropy parameters. Now the anchors, half of all the latent elements in $\hat{\boldsymbol{y}}$, can be decoded by AD. Then decoded anchors become visible for decoding non-anchors in the next pass.

14775

In the second decoding pass, the context feature for non-anchors can be calculated using proposed checkerboard model and entropy parameters of them are calculated from concatenated hyperprior feature and context feature. Then AD can decompress the remained half of $\hat{y}$ (those non-anchors) from the bitstream and we obtain whole latent $\hat{y}$. Finally, we get reconstructed image $\hat{x} = g_s(\hat{y})$.

### 4.3.3 Structure and Analysis

As an example, Figure 6 shows how we adapt the structure of the autoregressive model [26] by replacing its serial context model with proposed checkerboard context model. We don't modify other components of the autoregressive model for fair comparison.

For convolution kernel size of $k$, $\frac{k^2}{2}$ neighbors evenly distribute around the center (as is shown in Figure 1(c)) and contribute to the context for each non-anchor spatial location. Though only half of the latents refers to their neighbours now, a checkerboard context model can extract more causal information from decoded neighbours to save more bit rate. This compensates the potential BPP increase when compressing anchors without using any context in our proposed method.

Compared with the serial context model [26] which requires $H \times W$ sequential steps to decode a $H \times W \times M$ latent feature map, our proposed parallel model allows a constant step of 2 to decode such latents, where $H \times W \times \frac{M}{2}$ latent representations are processed in parallel. Considering that large-size images with more than one million pixels are produced and shared frequently nowadays, this parallelizable checkerboard context model brings significant improvement on practicality.

## 5. Experiments

We implement representative previous works and our proposed methods in PyTorch [29]. We choose the largest 8000 images from the ImageNet [12] validation set as our training data, where each image has more than one million pixels. Following previous works [5, 6], we add random uniform noise to each of them and then downsample all the images. We use Kodak dataset [17] and Tecnick dataset [2] as our test set.

During training, before fed into models all input images are randomly cropped to $256 \times 256$ patches. All models are trained for 2000 epochs (*i.e.* 2M steps) with a batch-size of 8 and learning rate of $10^{-4}$ if not specified. Sadam optimization [7] is adopted on each convolution layer for training stability.

### 5.1. Toy Experiments on Random-Mask Model

We adapt the architecture of Minnen2018 [26] by replacing its context model by a $5 \times 5$ random-mask convolution

| description | $K_{ref}$ | $R_M$ | $\eta(M)$ |
|---|---|---|---|
| non-reference ($R_0$) | 0 | 0.4332 | 0.0% |
| serial $3 \times 3$ (Fig. 1(a)) | 4 | 0.3928 | 9.3% |
| serial $5 \times 5$ (Fig. 1(b)) | 12 | 0.3817 | 11.9% |
| checkerboard $3 \times 3$ (Fig. 1(c)) | 4 | 0.3651 | 15.7% |
| checkerboard $5 \times 5$ (Fig. 1(d)) | 12 | 0.3648 | 15.8% |
| all neighbours in $3 \times 3$ | 8 | 0.3649 | 15.8% |

Table 1. Various masks tested with the random-mask model. $K_{ref}$ denotes the number of referred neighbours (*i.e.* number of ones in the mask). $R_M$ and $\eta(M)$ denote the BPP and rate saving ratio when feeding mask $M$ to the random-mask model. The row *all neighbours in* $3 \times 3$ refers to the mask where all 8 neighbours surrounding the center are set to 1. Note that the random-mask model is not designed as an actual codec, during decoding we can use entropy model parameters obtained in the encoding process, which is not allowed in actual decoder.

to obtain the random-mask model mentioned in section 4.1. Here we further discuss its details. To generate a binary mask, each location of the mask is sampled from $\{0, 1\}$ in equiprobability. During training, new random masks are generated in every iteration. To train such a model with random-mask context, we set $\lambda = 0.01$ and $N = M = 256$ for MSE optimization. On the trained model we perform two toy experiments:

- **Single Reference Mask for Causal Estimation**. As above mentioned, we estimate the causal relationship between latent pairs by using $5 \times 5$ masks with only one location set to 1 and calculating their $\eta$ on Kodak. Figure 4(a) shows the result which implies that the mutual information between latents decays quickly as the spatial distance between them increases.

- **Performance of Various Masks**. We test several masks using the random-mask model and post a brief report in Table 1. It indicates that simply increasing the number of referred neighbours (*i.e.* $K_{ref}$ in the table) doesn't always help save bit rate, because the pattern referring to 4 adjacent neighbours (the row *checkerboard* $3 \times 3$) performs much better than the pattern referring to 12 left-top neighbours (the row *serial* $5 \times 5$). Also, even referring to all 8 surrounding neighbours cannot outperform *checkerboard* $3 \times 3$. However, masks referring to more adjacent neighbours (the last three rows) always perform better. This further proves that closer neighbours play much more important roles in context modeling. Motivated by this experiment, we establish our parallel context model based on *checkerboard* $5 \times 5$.
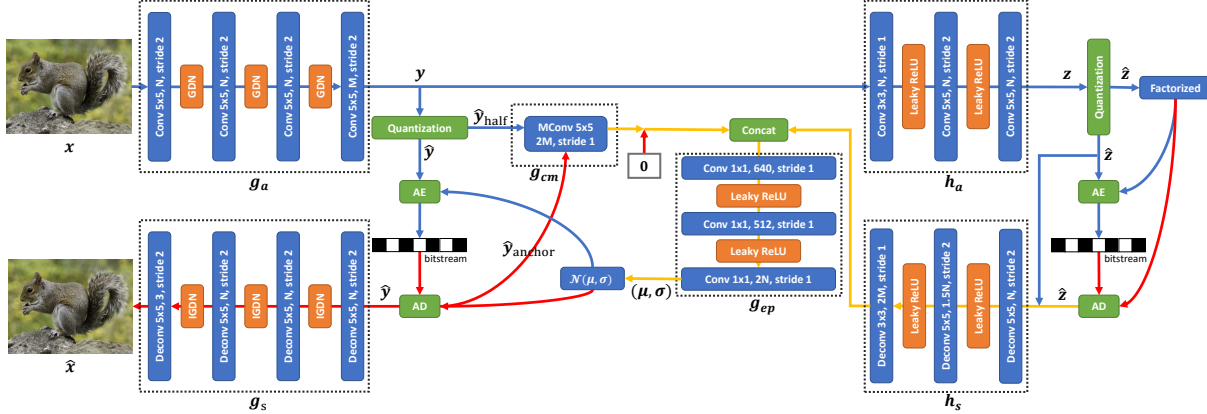
Figure 6. Previously proposed framework of learned compression model using context based autoregressive entropy model [26]. We replace the serial context model with the proposed checkerboard convolution. Blue and red lines denote encoding and decoding data flow respectively. Processes shared by both encoding and decoding are represented by yellow lines.

| architecture (N=192) | Ballé2018 | Minnen2018 | | | Cheng2020 | |
|---|---|---|---|---|---|---|
| | | w/o context | serial | parallel(ours) | serial | parallel(ours) |
| Kodak (768 × 512) | 26.34 | 26.41 | 1323.66 | 29.66 | 1395.35 | 75.23 |
| Tecnick (1200 × 1200) | 83.28 | 86.31 | 4977.98 | 95.46 | 5296.16 | 259.37 |

Table 2. Total decoding time averaged on Kodak and Tecnick (ms). Feature map size of each Kodak image is $48 \times 32 \times M$ and feature map size of each Tecnick image is $75 \times 75 \times M$. Ballé2018 is the earlier context-free hyperprior model [6] while Minnen2018 and Cheng2020 represent the serial autoregressive structure [26] and its Gaussian Mixture Model (GMM) and attention involved adaption [10] respectively. For models marked as *serial* the serial context models are adopted and for *parallel* ones the proposed checkerboard context is used.
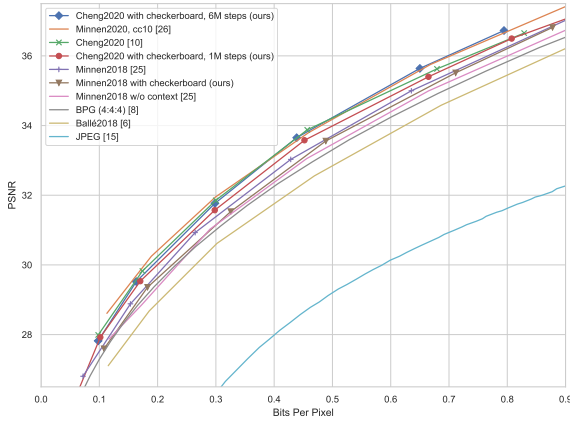


Figure 7. RD curves of various learned or manually designed image codecs. The results are averaged on Kodak. Except for two models with our proposed context model, all data are reported by prior works. All learned models are optimized for MSE. For MS-SSIM optimization results please refer to the supplementary material.
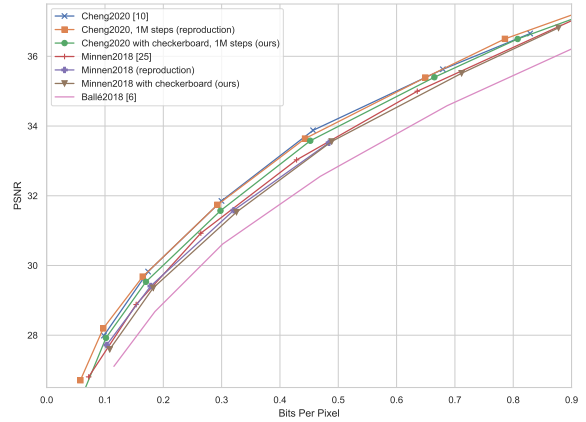


Figure 8. Comparison of parallel and serial context models on Minnen2018 and Cheng2020. Curves marked as *ours* and *reproduction* are trained on our dataset using above specified settings, while the rest are reported results.

## 5.2. Checkerboard Context Model V.S. Serial Context Model

We evaluate our checkerboard context model and the parallel decoding method based on two previous architectures using serial context: Minnen2018 [26] and Cheng2020 [10]. As is explained in section 4.3.3 and shown in Figure 6 using

Minnen2018 as an example, we do not change any other architectures except replacing the serial context model by the proposed one to ensure fair comparisons.

To compare the performance of different context models, we implement these two baseline models, and for each baseline we train its adaption with parallel context model under the same settings. The detailed experimental settings are:

| Minnen2018 (N=192) | serial | parallel(ours) |
|---|---|---|
| hyper synthesis | 1.26 | 1.42 |
| parameter calculation | 1302.42 | 4.75 |
| latent synthesis | 20.98 | 23.49 |
| total | 1323.66 | 29.66 |

Table 3. Running speed of Minnen2018's each decoding process (ms). In *hyper synthesis* and *latent synthesis*, the decoders $h_s$ and $g_s$ are invoked respectively. In *parameter calculation*, context features and entropy parameters are calculated by $g_{cm}$ and $g_{ep}$.

- **Cheng2020**. We train Cheng2020 following their reported settings, *i.e.* K = 3, $\lambda$ = {0.0016, 0.0032, 0.0075, 0.015, 0.03, 0.045}, N = M = 128 for the three lower $\lambda$ and N = M = 192 for the rest[1]. For reproduction we train 1M steps, and we find that training up to 6M steps can lead to an even better RD performance especially at high bit rates. During training, after 3M steps we decay the learning rate from $10^{-4}$ to $5 \times 10^{-5}$.

- **Minnen2018**. As the exact setting was not reported, we choose to use the same $\lambda$ setting as Cheng2020. We follow the suggestion on reproduction from authors[2] to use N = 128 and M = 192 for small $\lambda$ and N = 192 and M = 320 for big $\lambda$ and consider $\lambda \geq 0.015$ (corresponding BPP > 0.6) as big ones. For all models we train 6M steps and the learning rate decays to $5 \times 10^{-5}$ after 3M steps.

### 5.2.1 Decoding Speed

We evaluate inference latency of both serial and parallel models on Nvidia TITAN XP: Table 2 and Table 3. Notice that for proposed parallel models, the context model and the parameter network are invoked twice in the parameter calculation process. For serial models, context model and parameter network are called $H \times W$ times for decoding a $H \times W \times M$ feature map. For each time we crop the visible neighbours to $5 \times 5$ patches then feed them into the serial context model for a fast implementation [38] to pursue a fair comparison. It is obvious that a serial context model becomes a bottleneck even when decoding relatively small images. Meanwhile, the proposed two-pass decoding model has a much faster running speed on parallel devices, making spatial context based image compression models more practical.

### 5.2.2 Rate-Distortion Performance

We train and compare proposed checkerboard context model and the serial model based on Minnen2018 and

---

[1]The hyper-parameter M is not introduced in the original paper. Here we let M denote number of output channel of the encoder $g_a$'s last layer.

[2]https://groups.google.com/g/tensorflow-compression/c/LQtTAo6l26U/m/cD4ZzmJUAgAJ

Cheng2020 with settings as mentioned above. Figure 8 shows their RD-curves evaluated on Kodak (the ones marked as *with checkerboard*). Compared with original models, using such a parallel context model only slightly reduces RD performance on Kodak. However, it still outperforms the context-free hyperprior baseline (Ballé2018 in the figure) significantly (compared with Ballé2018, BDBR $-17.0\%/-27.4\%$ for Minnen2018/Cheng2020 using our checkerboard context model). Since it removes the limitation of computational efficiency, we think the quality loss is acceptable.

For completeness, we also compared our proposed models with several previous learned or conventional codecs including a recently proposed channel conditioned model [27], a SOTA architecture without using spatial context. See Figure 7, the channel conditioned model performs better than Cheng2020 with a parallel context model at lower bit rate but slightly worse at higher bit rate. As discussed by the author, a potential combination of spatial context and channel-wise adaption is promising to further improve RD performance of present approaches, since only the spatial-wise adaption can help remove spatial redundancy from source images. However, investigating the proper way of combining the two different type of adaptive modeling is beyond the scope of this paper as our topic is to speed up the spatial context model. To our understanding, the proposed parallel and efficient spatial context model is an important basis for future researches on the multi-dimension context modeling techniques, or the running speed will be inevitably limited by the low-efficiency serial context model.

We have also tested all above-mentioned models on Tecnick dataset and come to the same conclusion that the proposed checkerboard context model can be used as an efficient replacement to the serial context. For more RD curves and results please refer to the supplementary material.

## 6. Discussion

Serial context models for learned image compression are computationally inefficient. We propose a parallel context model based on the checkerboard-shaped convolution and develop a two-pass parallel decoding scheme. Compared with the serial context model, it allows decoding to be implemented in a highly parallel manner. After applying it to two representative context model involved baselines, we prove that it greatly speeds up the decoding process on parallel devices while keeps a competitive compression performance. Also, our proposed approach does not require any changes in model structure or model capacity, so it is almost a drop-in replacement for the original widely used serial context model. Therefore, the proposed approach significantly improves the potential performance of SOTA learned image compression techniques with spatial context models.

# References

[1] Eirikur Agustsson, Michael Tschannen, Fabian Mentzer, Radu Timofte, and Luc Van Gool. Generative adversarial networks for extreme learned image compression. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 221–231, 2019.

[2] N. Asuni and A. Giachetti. Testimages: a large-scale archive for testing visual devices and basic image processing algorithms. In *STAG: Smart Tools & Apps for Graphics (2014)*, 2014.

[3] Mohammad Haris Baig, Vladlen Koltun, and Lorenzo Torresani. Learning to inpaint for image compression. In *Advances in Neural Information Processing Systems*, pages 1246–1255, 2017.

[4] Johannes Ballé, Nick Johnston, and David Minnen. Integer networks for data compression with latent-variable models. In *Int. Conf. on Learning Representations*, 2019.

[5] Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. In *Int. Conf. on Learning Representations*, 2017.

[6] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. In *Int. Conf. on Learning Representations*, 2018.

[7] J. Ballé. Efficient nonlinear transforms for lossy image compression. In *2018 Picture Coding Symposium (PCS)*, pages 248–252, June 2018.

[8] Fabrice Bellard. Bpg image format. *URL https://bellard.org/bpg*, 2015.

[9] B Bross, J Chen, and S Liu. Versatile video coding (draft 5). *JVET*, 2019.

[10] 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 (CVPR)*, June 2020.

[11] Yoojin Choi, Mostafa El-Khamy, and Jungwon Lee. Variable rate deep image compression with a conditional autoencoder. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3146–3154, 2019.

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[13] Alexey Dosovitskiy and Djolonga Josip. You only train once: Loss-conditional training of deep networks. In *Int. Conf. on Learning Representations*, 2020.

[14] V. K. Goyal. Theoretical foundations of transform coding. *IEEE Signal Processing Magazine*, 18(5):9–21, Sep. 2001.

[15] ITU. Information technology - digital compression and coding of continuous - tone still images - requirements and guidelines. *CCITT, Recommendation*, 1992.

[16] Nick Johnston, Elad Eban, Ariel Gordon, and Johannes Ballé. Computationally efficient neural image compression. *arXiv preprint arXiv:1912.08771*, 2019.

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

[18] Jooyoung Lee, Seunghyun Cho, and Seung-Kwon Beack. Context-adaptive entropy model for end-to-end optimized image compression. In *Int. Conf. on Learning Representations*, 2019.

[19] Jooyoung Lee, Seunghyun Cho, Se-Yoon Jeong, Hyoungjin Kwon, Hyunsuk Ko, Hui Yong Kim, and Jin Soo Choi. Extended end-to-end optimized image compression method based on a context-adaptive entropy model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[20] Jooyoung Lee, Seunghyun Cho, and Munchurl Kim. An end-to-end joint learning scheme of image compression and quality enhancement with improved entropy minimization. *arXiv preprint arXiv:1912.12817*, 2019.

[21] Mu Li, Kede Ma, Jane You, David Zhang, and Wangmeng Zuo. Efficient and effective context-based convolutional entropy modeling for image compression. *IEEE Transactions on Image Processing*, 29:5900–5911, 2020.

[22] Mu Li, Wangmeng Zuo, Shuhang Gu, Debin Zhao, and David Zhang. Learning convolutional networks for content-weighted image compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3214–3223, 2018.

[23] Detlev Marpe, Heiko Schwarz, and Thomas Wiegand. Context-based adaptive binary arithmetic coding in the h. 264/avc video compression standard. *IEEE Transactions on circuits and systems for video technology*, 13(7):620–636, 2003.

[24] G Martin. Range encoding: an algorithm for removing redundancy from a digitised message. In *Video and Data Recording Conference, Southampton, 1979*, pages 24–27, 1979.

[25] Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Conditional probability models for deep image compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4394–4402, 2018.

[26] David Minnen, Johannes Ballé, and George D Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *Advances in Neural Information Processing Systems*, pages 10771–10780, 2018.

[27] David Minnen and Saurabh Singh. Channel-wise autoregressive entropy models for learned image compression. 2020.

[28] David Minnen, George Toderici, Saurabh Singh, Sung Jin Hwang, and Michele Covell. Image-dependent local entropy models for learned image compression. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 430–434. IEEE, 2018.

[29] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

[30] Majid Rabbani. Jpeg2000: Image compression fundamentals, standards and practice. *Journal of Electronic Imaging*, 11(2):286, 2002.

[31] Oren Rippel and Lubomir Bourdev. Real-time adaptive image compression. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2922–2930. JMLR. org, 2017.

[32] Jorma Rissanen and Glen Langdon. Universal modeling and coding. *IEEE Transactions on Information Theory*, 27(1):12–23, 1981.

[33] David Slepian and Jack Wolf. Noiseless coding of correlated information sources. *IEEE Transactions on information Theory*, 19(4):471–480, 1973.

[34] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy image compression with compressive autoencoders. In *Int. Conf. on Learning Representations*, 2017.

[35] George Toderici, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor, and Michele Covell. Full resolution image compression with recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5306–5314, 2017.

[36] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multi-scale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003.

[37] Sihan Wen, Jing Zhou, Akira Nakagawa, Kimihiko Kazui, and Zhiming Tan. Variational autoencoder based image compression with pyramidal features and context entropy model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[38] Jing Zhou, Sihan Wen, Akira Nakagawa, Kimihiko Kazui, and Zhiming Tan. Multi-scale and context-adaptive entropy model for image compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[39] Lei Zhou, Chunlei Cai, Yue Gao, Sanbao Su, and Junmin Wu. Variational autoencoder for low bit-rate image compression. In *CVPR Workshops*, pages 2617–2620, 2018.

[40] Lei Zhou, Zhenhong Sun, Xiangji Wu, and Junmin Wu. End-to-end optimized image compression with attention mechanism. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.