A Spatial RNN Codec for End-To-End Image Compression

Chaoyi Lin, Jiabao Yao, Fangdong Chen, Li Wang
Hikvision Research Institute
Hangzhou, China
{linchaoyi, yaojiabao, chenfangdong, wangli7}@hikvision.com

Abstract

Recently, deep learning has been explored as a promising direction for image compression. Removing the spatial redundancy of the image is crucial for image compression and most learning based methods focus on removing the redundancy between adjacent pixels. Intuitively, to explore larger pixel range beyond adjacent pixel is beneficial for removing the redundancy. In this paper, we propose a fast yet effective method for end-to-end image compression by incorporating a novel spatial recurrent neural network. Block based LSTM is utilized to remove the redundant information between adjacent pixels and blocks. Besides, the proposed method is a potential efficient system that parallel computation on individual blocks is possible. Experimental results demonstrate that the proposed model outperforms state-of-the-art traditional image compression standards and learning based image compression models in terms of both PSNR and MS-SSIM metrics. It provides a 26.73% bits-saving than High Efficiency Video Coding (HEVC), which is the current official state-of-the-art video codec.

1. Introduction

Image compression is an important technique for reducing communication traffic and saving data storage. Most traditional lossy image compression standards such as JPEG [25], WebP [4] and Better Portable Graphics (BPG) [5] are based on transform coding [9] framework. In this framework, a prediction transform module is used to map image pixel into a quantized latent representation and then compress the latents by entropy coding.

Recently, the deep neural networks (DNNs) have shown their great advantages in various areas. Along with this progress of deep learning, learning based image compression models also have derived significant interests [14, 8, 19, 17, 3, 18, 10, 1, 11]. Auto-encoder is usually applied in image compression that an encoder transforms input image into a latent representation, and the decoder inversely transforms a quantized latent representation into the reconstruction of input image. The neural networks in auto-encoder approximate nonlinear functions, which can map pixels into a more compressible latent space than the linear transform used by traditional image compression standards. Another advantage of learning based image compression models is that they can be easily optimized for specific metric such as SSIM [26] and MS-SSIM [27] by changing the loss function.

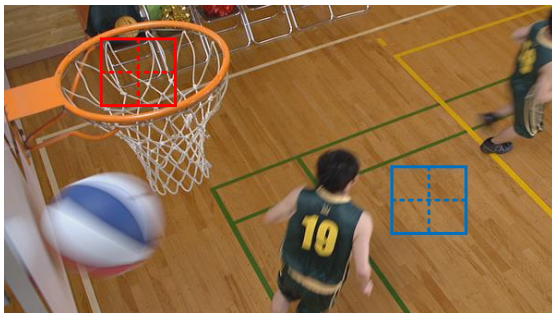


Figure 1. The effective of block based methods. In the blue region, both the correlation for adjacent pixels and adjacent blocks are large. In the red region, due to the similar texture in different blocks, the correlation between adjacent blocks are larger than that between adjacent pixels.

Very recently, a few learning based image compression models have outperformed the state-of-the-art traditional image compression standard BPG in terms of PSNR metric [28, 13, 18, 7]. These works focus on removing the redundant information between adjacent pixels by CNN. However, in the latest developing image/video compression standards, such as Versatile Video Coding (VVC) [6], block based processing is preferred. By using the block based processing, the redundant information for both adjacent pixels and blocks can be removed through block based prediction transform [15]. Figure 1 illustrates the effective of block based methods. High correlation for adjacent pixels and blocks can be found in the region marked by blue line. In this case, both pixel based methods and block based methods are effective. However, in the red region, the pixel

based methods can barely capture the redundancy because correlation for adjacent pixels is low. By using block based methods, the similar textures can be found between adjacent blocks and the spatial redundancy can be removed effectively in this case. This demonstrates that block based methods can further improve compression performance. However, it is seldom explored in learning based image compression model.

Inspired by the latest compression standards, we propose a spatial RNN architecture for lossy image compression model. The spatial RNN architecture fully exploits spatial correlations existing in adjacent blocks through block based LSTM, which can further remove spatial redundant information. Besides, the adaptive quantization is adopted in our model where the network would learn to automatically allocate bits for the latent map according to its contents. Moreover, two hyperprior network are adopted instead of context model in proposed entropy model by considering both the performance and efficiency. Experimental results demonstrate that proposed image compression model can outperform state-of-the-art traditional compression standards BPG and other deep learning based image compression models. Moreover, proposed method is potential for parallel computing which is highly efficient.

2. Related work

Many standard codecs have been developed for lossy image compression. The most widely used lossy compression standard is JPEG. More sophisticated standards such as WebP and BPG are developed to be portable and more compression-efficient than JPEG. To our knowledge, BPG has the highest compression performance among existing lossy image compression standards.

Recently, applying neural network to image compression has attracted considerable attention. Neural network architecture for image compression are usually based on auto-encoder framework. In this framework, both recurrent neural network (RNN) [5, 24, 28] and convolution neural network (CNN) [1, 17, 2, 22, 10] based models have been developed. Toderici *et al*. [5] propose RNN architecture for variable-rate image compression framework which compresses a 32x32 image in a progressive manner. In [24], a general architecture for compressing full resolution image with RNN, residual scaling and a variation of gated recurrent unit (GRU) is presented. Weber *et al*. [28] utilize RNN based architecture for image compression and classification. Different with works [5, 24], which only focus on removing the redundant information within each block, the redundancy between adjacent blocks is explored in our block based LSTM recurrent network.

Entropy model, which approximates the distribution of discrete latent representation, improves the image compression performance significantly. Thus, recent methods have given increasing focus to entropy model to improve compression performance. Ballé *et al*. [3] propose to use hyperprior to effectively capture the spatial dependencies in the latent representation. They model the distribution of latent representation as a zero-mean Gaussian distribution with standard deviation σ. A scale hyperprior is introduced to estimate the σ by stacking another auto-encoder on latent representation. Minnen *et al*. [18] further utilize the hyperprior to estimate the mean and standard deviation of learned latent representation to help removing spatial dependencies from the latents. Besides, context model is adopted in their model for achieving higher compression rate and it is the first learning based model that outperform BPG on PSNR metric. Lee *et al*. [13] also represent a lossy image compression with context model and a parametric model for an entropy model of hyperprior. In above works, only one hyperprior network is used to estimate the entropy parameter μ and σ. However, in the proposed model, we find that using two joint hyperprior networks to estimate the entropy parameter respectively can further improve compression performance. Besides, though context model can improve the performance, it is time-consuming during decoding process. Thus, context model is not included in the proposed model for achieving lower computational complexity.

3. Proposed method

3.1. Overall framework

The overall framework is shown in Figure 2, where encoder E, decoder D, quantization net Q^z, and hyperprior networks E^{h^1}, D^{h^1}, E^{h^2}, D^{h^2} are neural networks. The proposed method incorporates analysis and synthesis transform, adaptive quantization and entropy model. The analysis transform generates the latent representation of the raw image while the synthesis transform maps the quantized latent back to reconstructed image. Firstly, the analysis transform E maps a block of one image x to the latent representation z. It is in this module that most spatial redundancy is removed. The quantization network Q^z generates the quantization steps s adaptively, which is then quantized to form the quantized latent $\hat{z} = Q(z; s)$. To achieve a higher compression performance, the latent is modeled as Gaussian distribution in our entropy model and two hyperprior networks are used to estimate the entropy parameters mean m and variance v of the distribution, respectively. Encoder then uses estimated entropy parameters to compress and transmit the quantized latent representation $\hat{z}$. It is worth noting that quantization steps s, quantized hyperprior $\hat{h}_1$, and $\hat{h}_2$ are also transmitted as side information. On the decoder side, The quantization steps s is first recovered to decode the hyperprior $\hat{h}_1$ and $\hat{h}_2$. The two hyperpriors are used to estimate the entropy parameters and then the estimated entropy parameters are utilized to recover the quan-

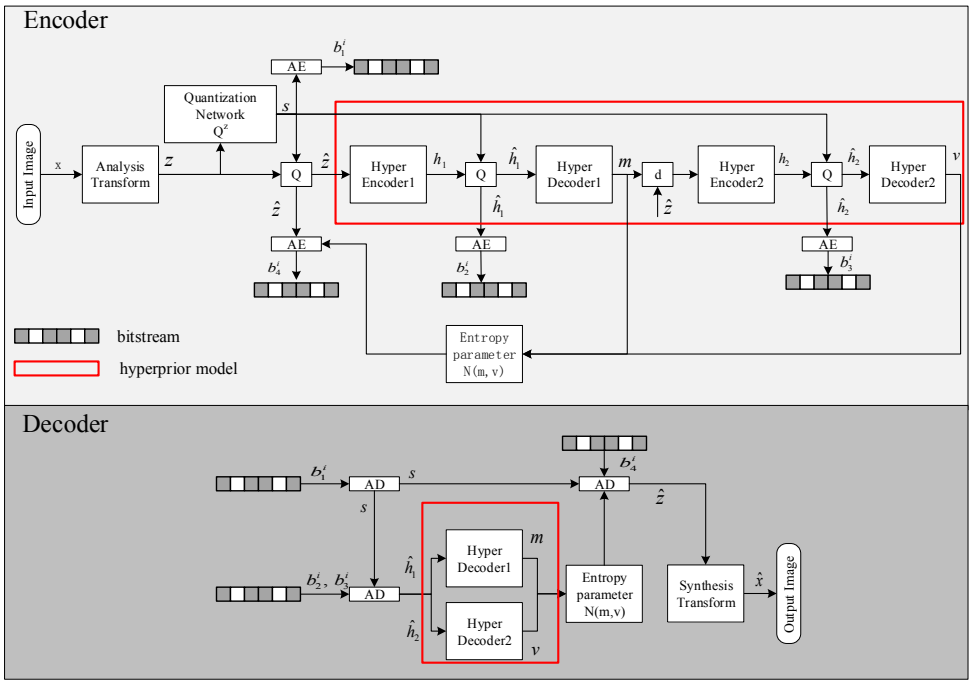


Figure 2. Network architecture of the proposed method. Q^z represents the quantization net, Q represents the quantization operation. z represents the full-precision latent representation of x, s represents the quantization step of Q , $\hat{z}$ is the integer-precision value of z/s. AE and AD represent the arithmetic encoder and arithmetic decoder respectively. $\hat{h}_1$ and $\hat{h}_2$ represent the quantized latent representation of the mean value m and variance value v of the Gaussian probabilistic density model N, d represents the pixel wise subtraction.

tized $\hat{z}$. Finally, synthesis transform maps the latent into the reconstructed image $\hat{x}$.

3.2. Analysis and synthesis transform

To take full use of the adjacent blocks to reduce the dependency between the blocks, we propose the block based Long Short-Term Memory (LSTM) architecture for image compression.

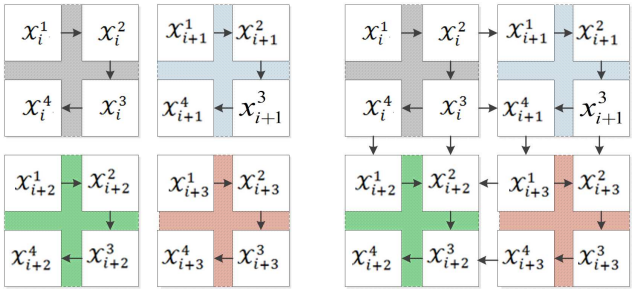
The Figure 3 (a) presents the proposed individual block based RNN (BRNN) process for removing the redundancy between the adjacent sub-blocks. The input image is firstly divided into non-overlapping blocks χ. The i-th block χ_i is then split into four sub-blocks $\chi_i^T : \{\chi_i^t, \chi_i^{t+1}, \chi_i^{t+2}, \chi_i^{t+3}\}$ with the size of $h \times w$ for temporary processing. The LSTM is used to process these sub-blocks recurrently in the order of: $TopLeft \rightarrow TopRight \rightarrow BottomRight \rightarrow BottomLeft$. The redundancy between sub-blocks can be removed in this recurrent process since the previous sub-block is used as the reference of current sub-block. In addition to the redundancy removing, each block χ_i is mapped to the latent representation individually, which demonstrates the potential to parallel computation for accelerating the encoding and decoding process.

Figure 3 (b) shows the highly recurrent RNN (HRNN) process to explore the redundancy removing between adjacent blocks. It is similar to the process of (a) except that there are some sub-blocks χ_i^{t+1} take the state of previous sub-blocks $\chi_{i-m}^{t_n}(0 < m < 4, 0 < n < 4)$ as input of hidden state. For example, sub-block χ_i^2 is the input of χ_{i+1}^1 which lies in the second blockχ_{i+1}. This methods can further exploit the correlation between adjacent blocks and achieve a higher performance. However, in this method, each block can not be computed concurrently and thus it is slower than method described in (a). Based on above analysis, method (a) is adopted in our model.

Let the split sub-block χ_i^t denotes the input, hid^t and c^t denote the hidden and cell states, respectively. After passing LSTM layer, the new hidden state hid^{t+1}, cell c^{t+1} are computed as:

$$\begin{aligned} \{o^t, f^t, i^t, g^t\} &= act(W_s \otimes hid^t + W_i \otimes x^t) \\ c^{t+1} &= f^t \odot c^t + i^t \odot g^t \\ hid^{t+1} &= o^t \odot tanh(c^t) \end{aligned} \tag{1}$$

where $\otimes$ and $\odot$ represent the convolution and element-wise multiplication operation, W_i and W_s are the weights of convolution layer C_i and C_s for the input components and state-to-state gates, respectively. The o^t, f^t, i^t, g^t denotes the output gates, forget gates, input gates and contents gates, respectively. The $act()$ is the activation function, which is sigmoid function for o^t, f^t, i^t and is tanh function when referred to g^t.



(a) (b)

Figure 3. Visualization of the input-to-state and state-to-state mappings for the proposed partitions. The left shows the process of BRNN, the right shows the process of HRNN.

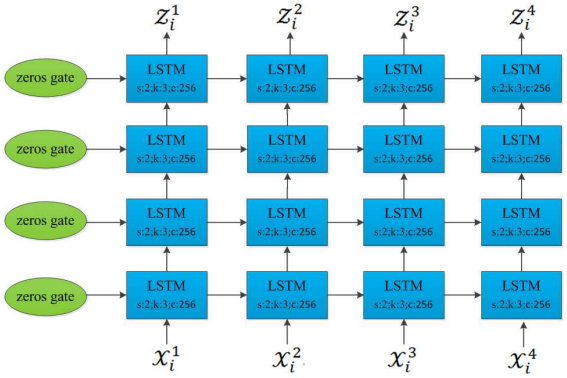


Figure 4. For LSTM in encoder side, the k represents the kernel size for both state-to-state convolutional layer C_s and input convolutional layer C_i, the c represents the output channel number of the hidden cell and output cell, the channel number of C_s and C_i must be set to the quadruple of c, the s represents the stride number of C_i. For LSTM in decoder side, the C_i is set to de-convolution operation with the up-sample factor s. For the last convolution layer in encoder, the channel number M is chosen based on the λ.

The layer details of encoder E and decoder D are shown in Figure 4. We denote C_i as the state-to-state convolutional layer and C_s is the input convolutional layer. The size of the tensor T_i and T_s, which is the output of C_i and C_s separately, is $4c \times \frac{h}{2} \times \frac{w}{2}$ (where c corresponds to the number of channels). After the activation operation, the tensor is split into 4 chunks which are the input of four gates, respectively. The size of each chunk is $c \times \frac{h}{2} \times \frac{w}{2}$. It should be noted that each sub-block χ_i^t shares the same weights of LSTM to ensure the invariance of the computed features. Finally, the output latents representation $Z\{z^t, z^{t+1}, z^{t+2}, z^{t+3}\}$ for each block are generated with the size of $4 \times M \times \frac{h}{16} \times \frac{w}{16}$ (M represents the output channel number of the last layer in encoder).

3.3. Quantization

It is found that great variability exists in the latents representation across channels, which means the importance of each channel should be different. Figure 5 shows the latent map for channel in a specific image. It can be seen that the first latent map preserves the high frequency characteristic since it preserves the details of the original images. Meanwhile, the last latent map represents the low frequency information. It can also be seen that the low frequency latent map is usually smooth and exists larger spatial redundancy than high frequency latent map. In practice, the smooth regions usually require less coding bits. Thus, less bits are allocated to the low frequency features by applying a larger quantization step. On the contrary, the high frequency features need a smaller quantization steps for achieving a better reconstruction quality. A quantization network is proposed to learn the quantization step adaptively in our model. As shown in the right side of Figure 5, the learned quantization steps are highly correlated with the latent feature maps.

Fabian *et al*. [17] use the importance map to allocate different regions with different amounts of bits. However, we train the quantization step s^i for each channel of the latent representation. The quantization step s^i can be obtained by the quantization net Q^z:

$$s^i = Q^z(\tilde{z}^i; \theta_q) \tag{2}$$

where the θ_q represents the weights of quantization network Q^z as shown in Figure 6.

The quantization of the latent representation z is a challenge for the end-to-end training, since the quantization operation is non-differentiable. Here we adopt Ballé's quantization operation [2] that the additive uniform noise is added to the latent during training to replace the non-differentiable quantization. This quantization is denoted by $\tilde{z}^i$. In the testing stage, actual quantization represented by $\hat{z}^i$ is used. The equations are shown as follow:

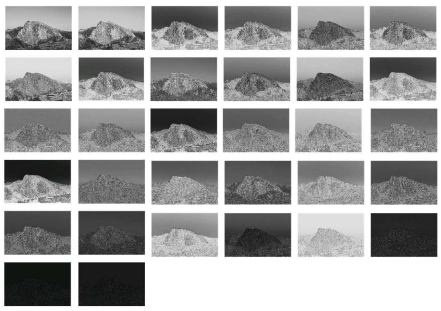
$$\begin{aligned} \tilde{z}^i &= Q(z^i, s^i) = z^i + \mu(-\frac{s^i}{2}, \frac{s^i}{2}) \\ \hat{z}^i &= Q(z^i, s^i) = \left\lfloor \frac{z^i}{s^i} + 0.5 \right\rfloor \times s^i \end{aligned} \tag{3}$$

where $\mu(-\frac{s^i}{2}, \frac{s^i}{2})$ represents the uniform noise which range from $-\frac{s^i}{2}$ to $\frac{s^i}{2}$, $\lfloor \cdot \rfloor$ represents the floor operation.

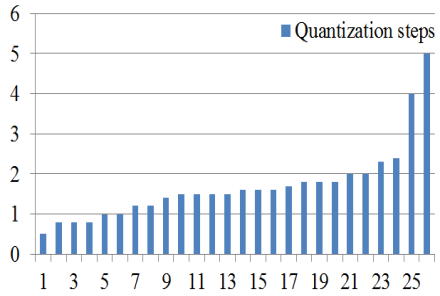
3.4. Advanced multiple correlated hyperprior model

To improve the efficiency and reduce the dependency of the context, we adopt the hyperprior model [3] to estimate the probability of the current element individually instead of the context-based method. Furthermore, we extend the hyperprior model by analyzing the correlation among the mean m^i, the variance σ^i of Gaussian probability density model and the latent representations z^i. Two experiments are designed to control each variable. Firstly, we fix the variance σ^i and quantization step s of each latent representations, and derive the optimal mean m^i for the latent z^i. Then we fix the mean m^i and quantization step s to derive

(a. The original image.)



(b. The latent maps.)



(c. The trained quantization steps.)

Figure 5. The visualization of the latent representation in each channel, and the average quantization step of the corresponding latent representation.

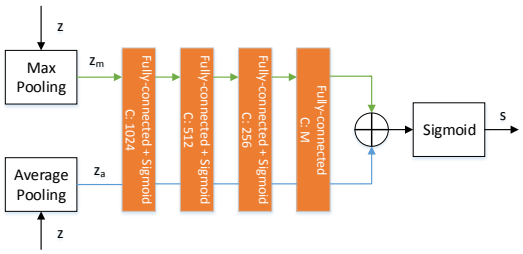


Figure 6. The architecture of Q^z. The weights of the fully connected layer are shared between z_m and z_a, which is inspired by [30]. The green arrows represent the data flow of z_m and the blue arrows represent the data flow of z_a. Each fully connected layer is followed by sigmoid function and the number of channels is denoted by C. For the last fully connected layer, the number of channels is equal to that of z.

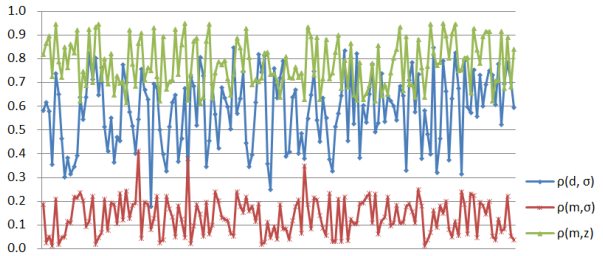


Figure 7. The correlation coefficient curve from training samples.

the optimal variance σ^i. The green line of Figure 7 presents the results of correlation coefficient between the m^i and the z^i, which indicates the latent representation and mean value are highly correlated. The red line represents the correlation coefficient between the m^i and the σ^i, which is less correlated than green line. Then we further analysis the correlation of σ^i and distance $d^i = z^i - m^i$. As shown in Figure 7, the σ^i is much more correlated with d^i than m^i. Therefore, different with previous works on estimating the entropy parameters of Gaussian distribution, the hyperprior is split into two sub-hyperpriors, one adopts the latent representation $\hat{z}^i$ as the input to estimate the mean value, the other estimates the σ^i based on the distance d^i.

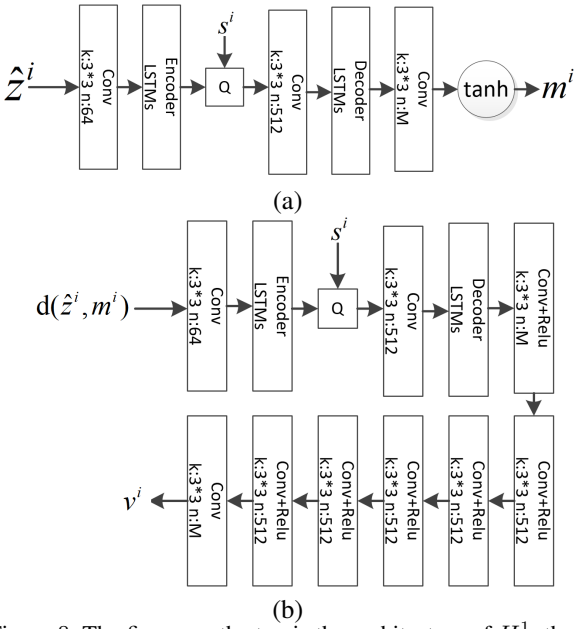


Figure 8. The figure on the top is the architecture of H^1, the bottom is the architecture of H^2. Q represents the quantization with the step s^i trained by Q^z. The Encoder LSTMs and Decoder LSTMs share all the weights of LSTM layers in encoder E and decoder D respectively.

We use two sub-modules H^1 and H^2, which generates mean m^i and variance v^i respectively to estimate the current latent value's probability. For the Gaussian distribution model, as the quantization step s is fixed, we can only get the max probability when $\hat{z}^i = m^i$. For H^1, it is taken as a data compression process same as encoder E and decoder D, therefore we share the same LSTM weights with them to save the memory allocation and training time. For H^2, the distance $d(\hat{z}^i, m^i)$ between $\hat{z}^i$ and m^i in element-wise is taken as the input of H^2 to generate the variance v^i. The probability mass functions can be described as:

$$p_{\hat{z}^i}(\hat{z}^i|m^i, v^i, s^i) = (N(m^i, v^i) * \mu(-\frac{s^i}{2}, \frac{s^i}{2}))(\hat{z}^i) \quad (4)$$

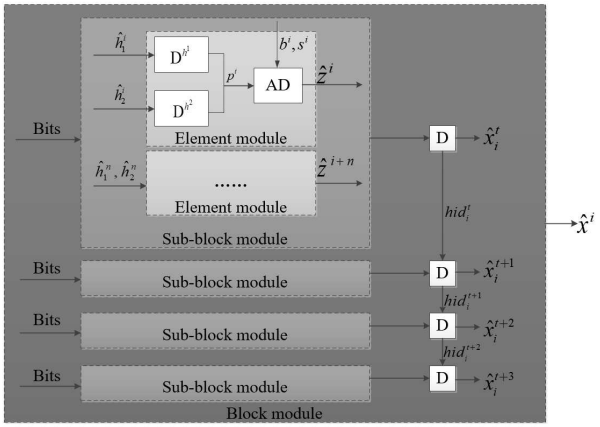


Figure 9. The parallel procedure in decoder side. AD represents the arithmetic decoding, D corresponds to Decoder, D^{h^1} and D^{h^2} represents the decoder of the two sub-hyperprior separately. The bits for each element module contain the side information $\hat{h}_1^i$, $\hat{h}_2^i$, s^i and the compact representation b^i. hid_i^t represents the hidden states of each block.

which can be evaluated in the closed form. Note that the s^i, $\hat{h}_1$ and $\hat{h}_2$ are all transmitted in bitstreams as the side information.

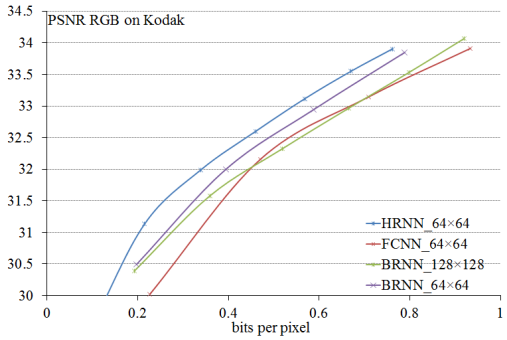


Figure 10. The RD curves of different models. The BRNN model is tested with the block size of 64×64 and 128×128, the HRNN model is tested with the block size of 64×64.

3.5. Parallel procedure

For the algorithm implementation on hardware, the long dependency between the elements is not desirable. For the encoder side, H^1 can only get started when the raw sub-block χ_i^t is encoded to the latent representation $\hat{z}^i$, and H^2 can only get started when the latent representation is encoded and decoded by H^1 to obtain m^i. For the decoder side, all the blocks χ_i^t and modules including Decoder D, the hyperprior decoder D^{h^1} and D^{h^1} can work concurrently. During the process of the arithmetic decoder, all elements can also work concurrently, since there is no dependency between them. Figure 9 shows the described parallel procedure in the decoder.

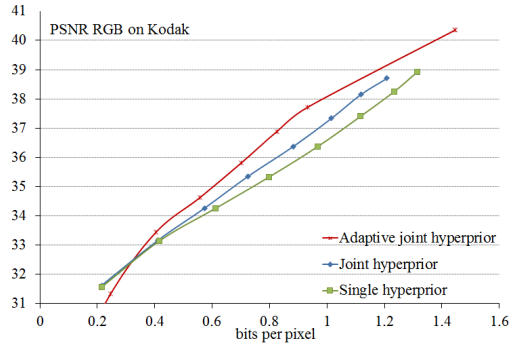


Figure 11. The RD curves of different hyperprior models.

3.6. Loss function

The goal of image compression is to generate a latent representation with the shortest bitstream which is evaluated by the entropy of quantized latent under a given distortion. The resulting objective function is shown in Equation 5:

$$L = \lambda \cdot d + r = \lambda \cdot \mathbb{E}_{x\sim p_x} d(x, \hat{x}) + \mathbb{E}_{x\sim p_x}[-log_2 p_{\hat{z}}(\hat{z})] + \mathbb{E}_{x\sim p_x}[-log_2 p_{\hat{h}_2}(\hat{h}_2)] + \mathbb{E}_{x\sim p_x}[-log_2 p_{\hat{h}_1}(\hat{h}_1)] \quad (5)$$

where d is the measurement between the reconstructed block $\hat{x}$ and the original block x, r denotes the costing of encoding $\hat{h}_1$, $\hat{h}_2$ and $\hat{z}$.The hyper-parameter λ controls the trade-off between the distortion d and the rates r in the loss function during training. λ is not a trainable variable, but a manually configured hyperparameter. It is worth noting that the rate of quantization step s is not included in Equation 5. We use context-based adaptive binary arithmetic coding (CABAC) in HEVC [29] to encode the quantization step, which is more compression-efficient. Our goal is to minimize Equation 5 over the training set χ by modeling the encoder E, decoder D, quantizer net Q^z and hyperprior E^{h^1}, D^{h^1}, E^{h^2}, D^{h^2}. In this paper, we use the mean square error (MSE) and MS-SSIM to measure the distortion.

4. Experiments

We use Pytorch as the training platform on NVIDIA Tian Xp with 12GB memory GPU. There are 30 thousands of images chosen from the DIV2K [23], ILSVRC2012 [21] in our training dataset. Before training, each image is cropped 128 × 128 pixels randomly. We use the Adam algorithm with a mini-batch size of 16 to train proposed models. The learning rate is fixed as 1e-4. The output channel number M of encoder is set to the integral multiple of 32, varying from 32 to 256. To enable comparison with other approaches, we present our performance on Kodak PhotoCD [12] dataset.

4.1. Spatial RNN architecture

In this section, different neural network architectures proposed in this paper are compared. We compare the

Table 1. The average time of different models with different block sizes on RGB Kodak.

Codecs	BRNN(128×128)	BRNN(64×64)	HRNN(64×64)	FCNN(128×128)
Encoding(ms)	8.719	10.304	27.214	2.519
Decoding(ms)	7.495	11.355	34.549	2.541

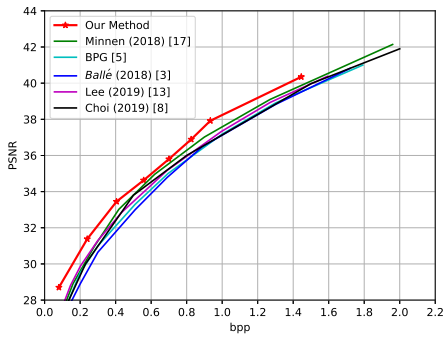


Figure 12. The RD curves of different methods for PSNR.

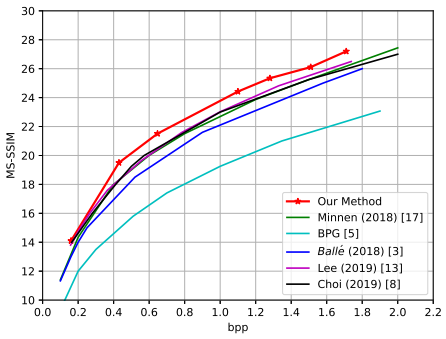


Figure 13. The RD curves of different methods for MS-SSIM.

Highly Recurrent HRNN, BRNN, and full convolution neural network (FCNN). The architecture of FCNN is the same with proposed network except that the RNN layers are replaced with convolution layers. Figure 10 shows the rate-distortion (RD) curves of different models tested on Kodak dataset. It can be seen that HRNN achieves the best performance. It is unsurprising because HRNN can fully use the information from adjacent blocks. The BRNN also performs well and FCNN performs the worst. This demonstrates that the RNN plays an important role in our model. Besides, we find that with the increase of the block size, the performance of BRNN tends to getting worse. This may mean that appropriate block size is important for image compression and we set block size as 128 in our model.

The Table 1 shows the average encoding and decoding time of different models with four threads for the block module. The FCNN is the fastest architecture among these models because no recurrent structure exists in FCNN. BRNN is also fast due to the parallelization between the blocks and we find that it becomes faster with increased block size. The HRNN takes the most encoding and decoding time compared to other architectures. HRNN is slow due to the dependency between adjacent blocks that blocks should be encoded or decoded consecutively, so the parallelization between the blocks can not be implemented in HRNN.

In general, although the HRNN shows the best performance in Figure 10, we adopt the BRNN with the block size of 128×128 in our model by considering the efficiency in practical application.

4.2. Performance results

Figure 12 and 13 show the RD curves of our BRNN model in terms of PSNR and MS-SSIM metrics, respectively. The state-of-the-art image codec BPG and other deep learning based models are compared with ou model. It can be seen from the results that our model not only outperforms the existing conventional state-of-the-art image codec BPG, but also the other deep learning based methods.

In BRNN process, multiple threads can be applied to improve the coding efficiency because the recurrent operation is performed within each individual blocks. To simulate the multiple threads operation, we divide input images into four non-overlapping sub-images and use four GPU cards to compress the sub-images. The average decoding time is compared with Minnen's method [3] in Table 3. As shown in Table 3, the proposed model is more efficient than Minnen [3] in decoder side.

To evaluate the improvement of our methods accurately, we compare our results with HEVC in BD-Rate metric [20]. The BD-Rate metric represents the average percent saving in bitrate between two methods, for a given objective quality. The configuration of HEVC is intra high throughput mode and is tested on HM16.14 [16] platform. As shown in Table 2, our method demonstrates 26.73% bits saving than HEVC. In Figure 14, we compare our subjective result to HEVC in low bitrate on kodim24 from Kodak. It can be seen that the output of our network has no obvious artifacts, even though our processing is block based and has no post-process filters like HEVC. The soft structures (like the texture on the wall) in our results are better preserved than HEVC . We refer to the supplementary material for further visual examples.

4.3. Ablation study

To verify the contribution of proposed hyperprior network and adaptive quantization network, we conduct the following ablation study. We train a image compression network with sigle hyperprior network, that only one sub-hyperprior network is used to estimate the mean and vari-

Table 2. The BD-Rate results compared with HM16.14.

		HM16.14		Our Method		BD-Rate
Images	QPISlice	Bits	PSNR	Bits	PSNR	RGB Average
Kodak	48	25594.00	25.79	159248.89	33.45	-26.73%
	44	49815.67	27.58	219397.43	34.63	
	40	93788.33	29.59	275256.94	35.80	
	36	168263.33	31.90	324618.71	36.89	
	32	283362.33	34.43	366991.17	37.92	
	28	452545.67	37.19	399079.66	38.76	
	24	688151.67	40.07	568386.18	40.35	

Table 3. The average times of different codec on RGB Kodak.

Codecs	Minnen *et al*. [3]	Our Method
Decoding(ms)	78124.802	28.421

ance of latent. The structure of this sigle hyperprior network is the same as hyperprior network H^1 described in section 3.4, except that the channel number of the last convolution layer in decoder side is twice of that of H^1 in order to obtain the estimated mean and variance. We also train the image compression model with proposed two sub-hyperprior networks, denoted as joint hyperprior. It should be noted that the adaptive quantization module is removed in above models. The proposed model with two sub-hyperprior networks and adaptive quantization, denoted as the adaptive joint hyperprior, is compared to the above two models. The RD curves of these models are shown in Figure 11. The results demonstrate that with adaptive quantization and joint hyperpriors, the results are much better than single hyperprior for high bitrate (greater than 0.3 bpp). However, it performs worse in low bitrate condition.

5. Conclusion

In this paper, we propose a novel spatial recurrent neural network for end-to-end image compression. The block based LSTM is utilized in spatial RNN to fully exploit spatial redundancy. Both the redundancy between adjacent pixels and blocks is removed. Besides, adaptive quantization step is adopted in our model which can automatically account for the trade-off between the entropy and the distortion. By considering both the performance and effeciency, two hyperprior network are adopted to replace context model in proposed entropy model. Experimental results show that proposed methods outperform the state-of-the-art methods, such as HEVC, BPG and other learning based compression method. Results on decoding time shows that proposed method is efficient and demonstrate the effectiveness of proposed parallel system.

References

[1] Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimization of nonlinear transform codes for perceptual quality. In *Picture Coding Symposium (PCS), 2016*, pages 1–5. IEEE, 2016. 1, 2

(a. Ours, 0.272bpp.)

(b. HEVC, 0.270bpp.)

(c. Original Kodak 24 image.)

Figure 14. The reconstruction from decoder generated by the adaptive joint hyperprior auto encoder and HEVC.

[2] Johannes Ballé, Valero Laparra, and Eero P Simoncelli.

End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704*, 2016. 2, 4

[3] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436*, 2018. 1, 2, 4, 7, 8

[4] Somnath Banerjee and Vikas Arora. Webp compression study." code. google. com/speed/webp/docs/webp study. html, 2011. 1

[5] Fabrice Bellard. Bpg image format (2017). *URL http://bellard. org/bpg/.[Online, Accessed 2016-08-05]*. 1, 2

[6] J Chen and E Alshina. Algorithm description for versatile video coding and test model 1 (vtm1). In *Document JVET-J1002 10th JVET Meeting*, 2016. 1

[7] 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. 1

[8] Gergely Flamich, Marton Havasi, and José Miguel Hernández-Lobato. Compression without quantization, 2020. 1

[9] Vivek K Goyal. Theoretical foundations of transform coding. *IEEE Signal Processing Magazine*, 18(5):9–21, 2001. 1

[10] Feng Jiang, Wen Tao, Shaohui Liu, Jie Ren, Xun Guo, and Debin Zhao. An end-to-end compression framework based on convolutional neural networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 2017. 1, 2

[11] Nick Johnston, Damien Vincent, David Minnen, Michele Covell, Saurabh Singh, Troy Chinen, Sung Jin Hwang, Joel Shor, and George Toderici. Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks. *structure*, 10:23, 2017. 1

[12] Eastman Kodak. Kodak lossless true color image suite (photocd pcd0992). *URL http://r0k. us/graphics/kodak*, 1993. 6

[13] Jooyoung Lee, Seunghyun Cho, and Seung-Kwon Beack. Context-adaptive entropy model for end-to-end optimized image compression. In *International Conference on Learning Representations*, 2019. 1, 2

[14] Xiang Li and Shihao Ji. Neural image compression and explanation, 2019. 1

[15] I. Matsuda, K. Unno, H. Aomori, and S. Itoh. Block-based spatio-temporal prediction for video coding. In *2010 18th European Signal Processing Conference*, pages 2052–2056, Aug 2010. 1

[16] K McCann, C Rosewarne, B Bross, M Naccari, K Sharman, and G Sullivan. High efficiency video coding (hevc) encoder description v16 (hm16). *JCT-VC High Efficiency Video Coding N*, 14:703, 2014. 7

[17] Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Conditional probability models for deep image compression. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 3, 2018. 1, 2, 4

[18] 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. 1, 2

[19] Ken Nakanishi, Shin ichi Maeda, Takeru Miyato, and Daisuke Okanohara. Neural multi-scale image compression, 2018. 1

[20] Stéphane Pateux and Joel Jung. An excel add-in for computing bjontegaard metric and its evolution. *ITU-T SG16 Q*, 6, 2007. 7

[21] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, and Michael Bernstein. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 6

[22] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy image compression with compressive autoencoders. *arXiv preprint arXiv:1703.00395*, 2017. 2

[23] Radu Timofte, Kyoung Mu Lee, Xintao Wang, Yapeng Tian, Ke Yu, Yulun Zhang, Shixiang Wu, Chao Dong, Liang Lin, and Yu Qiao. Ntire 2017 challenge on single image super-resolution: Methods and results. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1122–1131, 2017. 6

[24] 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 *CVPR*, pages 5435–5443, 2017. 2

[25] G. K. Wallace. The jpeg still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1), Feb 1992. 1

[26] Zhou Wang, Alan C Bovik, Hamid R Sheikh, Eero P Simoncelli, et al. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 1

[27] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003. 1

[28] Maurice Weber, Cedric Renggli, Helmut Grabner, and Ce Zhang. Lossy image compression with recurrent neural networks: from human perceived visual quality to classification accuracy. 2019. 1, 2

[29] Mathias Wien. High efficiency video coding. *Coding Tools and specification*, pages 133–160, 2015. 6

[30] Sanghyun Woo, Jongchan Park, Joon Young Lee, and In So Kweon. Cbam: Convolutional block attention module. 2018. 5