Subjective Quality Optimized Efficient Image Compression

Xining Wang†, Tong Chen†, and Zhan Ma‡
Vision Lab, Nanjing University
†{wangxn,tong}@smail.nju.edu.cn,‡mazhan@nju.edu.cn

Abstract

In this paper, we propose an efficient image compression framework that is optimized for subjective quality. Our framework is mainly based on the NLAIC (NonLocal Attention Optimized Image Coding) model which applied Variational Autoencoder (VAE) and non-local attention module to end-to-end image compression. This work makes two major contributions to the NLAIC framework. First, our models are optimized for subjective-friendly loss functions rather than conventional MSE (Mean Squared Error) or MS-SSIM (Multiscale Structural Similarity) which was widely used in previous works. Second, we introduce block-based inference mechanism to reduce the running memory consumption of the image compression network, and suggest a partial post-processing step to alleviate block artifacts caused by block-based inference in a lightweight computational fashion. Experiments have proved that the image reconstructed by our method can preserve more texture details than models trained for optimal MSE or MS-SSIM and also present capability for high-throughput decoding.

1. Introduction

Image coding, which is also known as image compression, refers to the idea that representing the image by a small number of bits under the condition of meeting certain quality evaluation. Traditional image coding methods like JPEG [12], JPEG2000 [9] and BPG [5] use linear transform or intra prediction to map the image to the transformed domain, and then perform quantization and entropy coding. With the rapid development of Deep Neural Networks (DNNs), learning-based image coding has improved the performance by introducing non-linear transform to image coding. Ballé *et al*. [3] applied the VAE to construct an end-to-end image compression framework, and further utilized a hyperprior module to generate more accurate probability estimate for latent features [4]. Recently, Chen *et al*. [6] introduced sparse non-local processing and 3-D masked convolutional neural network (CNN)-based context model into the VAE structure for optimized bit rate allocation and probability estimation.

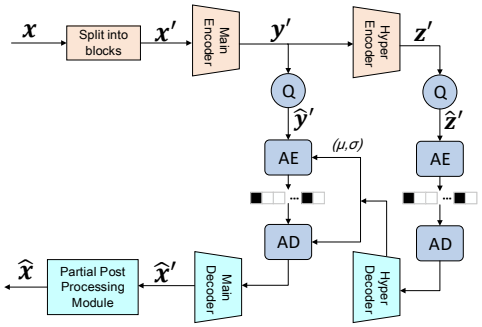
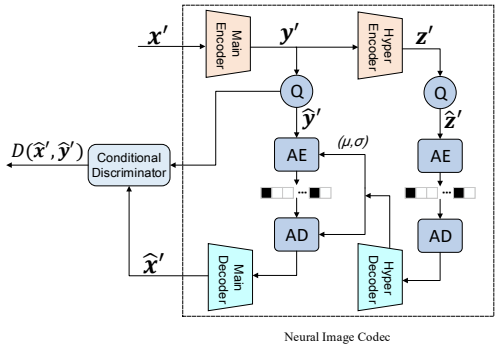


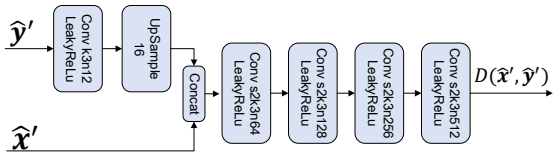
Figure 1: General Framework Illustration of Variational Autoencoder (VAE)-based Image Coding

Image compression methods mentioned above were mainly optimized for MSE or MS-SSIM [14], which failed to closely match the subjective quality of human perception. However, the key problem for image coding is to retain better subjective quality given the bit rate constraint because the human visual system (HVS) is the ultimate receiver for image content consumption. Recently, VGG-based [13] perceptual loss like DISTS [7] and LPIPS [16] and generative adversarial networks (GAN) made learning-based image compression codec more visually pleasing. For example, Mentzer *et al*. [11] and Agustsson *et al*. [2] followed the idea of rate-distortion-perception theory and used GAN to make reconstructed images more subjective-friendly.

In this work, we combine the idea of VAE-based and GAN-based image compression and propose a fast image codec with better subjective quality in reconstructed images. Unlike the usual way for training the VAE-based end-to-end image compression network, we train our codec in three steps, and in each step different modules are optimized by individual loss functions. Also, we propose block-based inference and a partial post-processing module. Our partial post-processing module can effectively reduce the block artifact caused by block-based inference with negligible computational overhead.

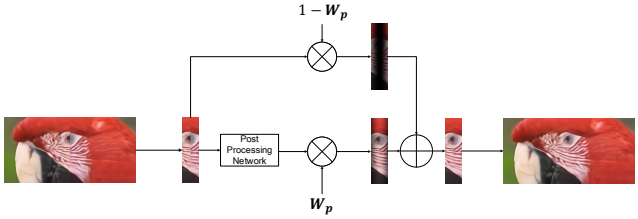


(a) Framework overview

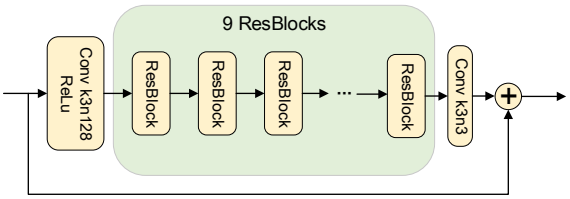


(b) The conditional discriminator module

Figure 2: Training framework.



(a) Partial post-processing process



(b) Structure of post-processing network

Figure 3: Partial post-processing.

2. Proposed Method

Figure 1 shows the framework of our method which mainly follows the idea of NLAIC [6] but removes the autoregressive context model to ensure fast decoding. We first split the input image $\boldsymbol{x}$ into blocks $\boldsymbol{x'}$ that can be encoded and decoded independently to avoid the large memory consumption that grows with the size of the input image. Each block in $\boldsymbol{x'}$ is transformed into latent features $\boldsymbol{y'}$ by main encoder and then put into the hyper encoder and decoder which generate the mean μ and variance σ to estimate the distribution $p(\hat{\boldsymbol{y}}'|\hat{\boldsymbol{z}}')$ for the quantized latent features $\hat{\boldsymbol{y}}'$. The output of the hyper encoder $\boldsymbol{z'}$ is quantized to $\hat{\boldsymbol{z}}'$ and encoded as the side information. On the decoding side, the main decoder transforms decoded $\hat{\boldsymbol{y}}'$ to reconstructed blocks $\hat{\boldsymbol{x}}'$ and the partial post-processing module puts the reconstructed blocks together and reduces block artifacts caused by block-based coding and decoding.

2.1. Training with Subjective-Friendly Loss

Our method combines the idea of VAE-based image codec with GAN-based image codec by proposing a three-stage training and uses different loss functions to train different parts of our image codec. First, we train the neural image codec which is shown in Figure 2 (a) with the commonly used Rate-Distortion (RD) loss function which is formulated in Eq. 1, where $d(\boldsymbol{x'}, \hat{\boldsymbol{x}}')$ denotes the distortion between the ground truth image and the reconstructed image output by the neural image codec. We use the weighted sum of MSE and the perceptual loss DISTS [7] as the distortion to ensure subjective quality while minimizing checkerboard artifacts when optimizing models for DISTS only, as formulated in Eq. 2. $r(\hat{\boldsymbol{y}}')$ means the estimated bitrate of $\hat{\boldsymbol{y}}'$ under the condition of $\hat{\boldsymbol{z}}'$ and $r(\hat{\boldsymbol{z}}')$ means the estimated bitrate of $\hat{\boldsymbol{z}}'$, denoted as Eq. 3 and Eq. 4 respectively.

$$L_{rd} = \lambda * d(\boldsymbol{x'}, \hat{\boldsymbol{x}}') + r(\hat{\boldsymbol{y}}') + r(\hat{\boldsymbol{z}}') \quad (1)$$

$$d(\boldsymbol{x'}, \hat{\boldsymbol{x}}') = w_1 * \mathbf{MSE}(\boldsymbol{x'}, \hat{\boldsymbol{x}}') + \mathbf{DISTS}(\boldsymbol{x'}, \hat{\boldsymbol{x}}') \quad (2)$$

$$r(\hat{\boldsymbol{y}}') = -\Sigma_i log_2(p_{\hat{y}'_i|\hat{z}'_i}(\hat{y}'_i|\hat{z}'_i)) \quad (3)$$

$$r(\hat{\boldsymbol{z}}') = -\Sigma_i log_2(p_{\hat{z}'_i}(\hat{z}'_i)) \quad (4)$$

In the second stage, we train the conditional discriminator [11], as illustrated in Figure 2 (b), with the loss function formulated below. Conv sakbnc means the convolution layer whose step size is a, kernel size is $b*b$ and the number of output channels is c. In this stage, we try to enable the discriminator to distinguish between the original image and the reconstructed image of the neural image codec.

$$L_D = -\mathbf{E}[log_2(1 - D(\hat{\boldsymbol{x}}', \hat{\boldsymbol{y}}')) - \mathbf{E}[log_2(D(\boldsymbol{x'}, \hat{\boldsymbol{y}}'))] \quad (5)$$

At last, we use the trained conditional discriminator to fine-tune our main decoder, just like the training of the generator when training the GAN-based image codec using the formulation below. We use not only the GAN loss generated by the conditional discriminator but also the weighted sum of GAN loss and DISTS to provide extra restriction on the finetuning of the main decoder.

$$L_{Dd} = -w_2 * \mathbf{E}[log_2(D(\hat{\boldsymbol{x}}', \hat{\boldsymbol{y}}'))] + \mathbf{DISTS}(\boldsymbol{x'}, \hat{\boldsymbol{x}}') \quad (6)$$

2.2. Partial Post-processing

Current learning-based post-processing methods usually feed the whole reconstructed image into the neural network

(a) (b) (c)

Figure 4: **The effect of the partial post-processing module.** (a) ground truth; (b) reconstruction w/o post-processing; (c) reconstruction w/ partial post-processing.

[15, 10] to avoid block artifacts in reconstructed images caused by block-based inference, which leads to huge but unnecessary calculation. In this section, we introduce a simplified *partial post-processing* module after the main decoder that has much lower computational complexity but can effectively reduce block artifacts.

Figure 3 (a) illustrates the process of partial post-processing. Rather than feed the whole reconstructed image into the network, our method only put the rectangular area near block boundaries into the post-processing network which is shown in Figure 3 (b), thus reducing calculations. To further reduce the discontinuity caused by directly replacing the boundaries with the output of the post-processing network, we perform a weighted summation of the output and input of the post-processing network. Here, we denote the pixels in the x^{th} column of the post-processing network's output by the post-processing network as $\boldsymbol{Y}(\boldsymbol{x})$ and the pixels in the x^{th} column of the post-processing network's input as $\boldsymbol{X}(\boldsymbol{x})$. $\boldsymbol{Y'}$ denotes the output after weighted summation, w denotes the width of processed boundaries and σ denotes the variance of the Gaussian-like weight. The whole process can be formulated as follows:

$$\boldsymbol{Y'}(\boldsymbol{x}) = \boldsymbol{Y}(\boldsymbol{x})W_p(x) + \boldsymbol{X}(\boldsymbol{x})(1 - W_p(x)) \quad (7)$$

$$W_p(x) = e^{-\frac{(x-\frac{w}{2})^2}{2\sigma^2}} \quad (8)$$

Figure 4 shows the effect of the partial post-processing module. It can be seen that reconstructed image with block artifacts which is shown in Figure 4 (b) has been significantly reduced after passing through our partial post-processing module.

3. Experimental Results

3.1. Experimental Conditions

Our network structure uses a simplified version of NLAIC [6] by removing the auto-regressive context model. Specifically, we set the number of channels of the convolutional layer in the main encoder and main decoder to 192 and set the number of channels of the convolutional layer in the hyper encoder and hyper decoder to 128. We use Pytorch and NVIDIA 1080Ti GPU for training and test, except for the memory consumption test which is performed on a 2-core CPU with 16G memory.

Table 1: Experimental results on CLIC2021 validation dataset.

bpp	0.075	0.148	0.266
PSNR (dB)	26.653	28.295	31.429
MS-SSIM	0.910	0.940	0.971
FID	194.696	183.518	155.656
Decoding Time (ours)	488s	583s	745s
Decoding Time [1]	5136s	6319s	7784s

1.averaged over top 10 submissions measured by FID in validation phase.

Images in the training dataset are cropped to the size of 256×256 and a total of 600k patches are generated for training. We first train the image codec for 15 epochs, setting w_1 as 100, then train the conditional discriminator for 2 epochs, finally set w_2 as 1 and fine-tune the main decoder for 2 epochs. Adam [8] optimizer is used. The initial learning rate is set to 5e-5 and cut by half every 3 epochs after the 7^{th} epoch. The learning rate is fixed to 5e-5 when training the conditional discriminator and finetuning the main decoder.

When training the partial post-processing module, the images in the training dataset are cropped into non-overlapping blocks of 128×128 pixels and coded/decoded independently by the deep neural codec. The reconstructed blocks are then spliced into images of 256×256 pixels as the training input of the post-processing network. The learning rate and the number of epochs are set as the same as the training parameters of the conditional discriminator. To simplify the training process, we use MS-SSIM for training the partial post-processing network rather than the perceptual loss functions mentioned above.

During test, we divide input images into non-overlapping blocks with a size of 2048×1024 pixels and set the width of partial post-processing w to 128 pixels.

3.2. Results

We use two images in Kodak dataset to show the subjective results of our method as shown in Figure 5. It can be seen that the reconstructed images don't perform well when using PSNR or MS-SSIM as the metric. However, by observing (c), (e) and (f), it can be seen that at similar or even lower bit rates, more details in the images are preserved such as the wall's texture in the pictures above and the grass in the pictures below. Figure 6 shows the RD curve tested on the Kodak [1] dataset, where the distortion is measured by FID using pytorch-fid (https:

Figure 5: **Experimental results using Kodak dataset**.(a) ground truth; (b) reconstructed images of high bitrate model; (c) reconstructed images of middle bitrate model; (d) reconstructed images of low bitrate model; (e) reconstructed images of MSE model; (f) reconstructed images of MS-SSIM model.

//github.com/mseitzer/pytorch-fid). We can see that our method outperforms NLAIC baselines optimized for MSE and MS-SSIM. Also, we test our method on the CLIC2021 dataset whose results are shown in Table 1. The results show that our method performs well in the FID metric with a much faster decoding speed compared with the top ten methods besides ours on the CLIC2021 validation leaderboard.

Table 2: Complexity comparison of full-image post-processing and partial post processing.

	FLOPs	Memory Consumption
Full-image Post-processing	15694G	Out of memory
Partial Post-processing	1473.84G	3314.113MB

Moreover, to compare the memory consumption and computational complexity between typical full-image post-processing and our proposed partial post-processing method, experiments are performed on images of 2K resolution (2048×1363) in the CLIC2021 validation datasets. It can be seen from Table 2 that our partial post-processing module saves over 10× FLOPs compared with full-image post-processing. Also, we can limit the memory consumption of post-processing to about 3GB, while for the full-image post-processing, it is even above the overall memory capacity of our test platform with 16GB memory.

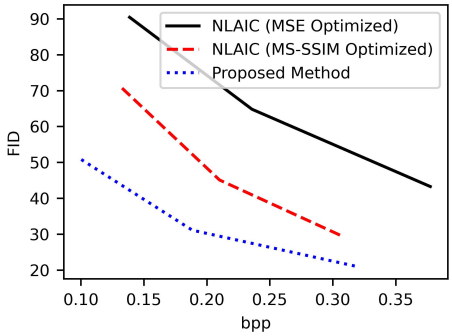


Figure 6: RD curve measured by FID.

4. Conclusion

In this paper, we brief the approach used for our CLIC2021 submission. Our method is designed on the basis of simplified VAE-based NLAIC and GAN-based image compression method. In addition, we propose a block-based inference and partial post-processing module to limit the memory consumption of image compression and reduce the blocking artifacts with less FLOPs and memory consumption. Experiments have proved that the proposed method can preserve more details of the original image in the reconstructed image. In the future, we will further analyze how different metrics would affect the distortion so as to design better quality metrics for subjective-friendly learned image compression codec.

References

[1] Kodak lossless true color image suite. http://r0k.us/graphics/kodak/, Dec. 15th 2016. 3

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

[3] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. End-to-end optimized image compression. In *International Conference on Learning Representations*, 2017. 1

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

[5] Better Portable Graphics. https://bellard.org/bpg/, 2018. 1

[6] T. Chen, H. Liu, Z. Ma, Q. Shen, X. Cao, and Y. Wang. End-to-end learnt image compression via non-local attention optimization and improved context modeling. *IEEE Transactions on Image Processing*, 30:3179–3191, 2021. 1, 2, 3

[7] Keyan Ding, Kede Ma, Shiqi Wang, and Eero P. Simoncelli. Image quality assessment: Unifying structure and texture similarity. *CoRR*, abs/2004.07728, 2020. 1, 2

[8] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980v9*, Jan. 2017. 3

[9] D. T. Lee. Jpeg 2000: Retrospective and new developments. *Proceedings of the IEEE*, 93(1):32–41, Jan 2005. 1

[10] 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. 3

[11] Fabian Mentzer, George D. Toderici, Michael Tschannen, and Eirikur Agustsson. High-fidelity generative image compression. In *Advances in Neural Information Processing Systems*, volume 33, pages 11913–11924, 2020. 1, 2

[12] Overview of JPEG. https://jpeg.org/jpeg/, 2018. 1

[13] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *Computer Science*, 2014. 1

[14] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Seventh Asilomar Conference on*, volume 2, pages 1398–1402. Ieee, 2003. 1

[15] Yaojun Wu, Xin Li, Zhizheng Zhang, Xin Jin, and Zhibo Chen. Learned block-based hybrid image compression. *arXiv preprint arXiv:2012.09550*, 2020. 3

[16] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 1