# Variational Autoencoder Based Image Compression with Pyramidal Features and Context Entropy Model

Sihan Wen*[1], Jing Zhou[1], Akira Nakagawa[2], Kimihiko Kazui[2], and Zhiming Tan[1]

[1]Fujitsu R&D Center Co. Ltd., [2]Fujitsu Laboratories Ltd.

[1]{wensihan, zhoujing, zhmtan}@cn.fujitsu.com,[2]{anaka, kazui.kimihiko}@fujitsu.com

## Abstract

*Variational autoencoder with the potential to address an increasing need for flexible lossy image compression, has recently be investigated as a promising direction for advancing the state-of-the-art. Based on this effective framework, we present an end-to-end image compression method with a multi-scale encoder, residual decoder, and separate entropy model. The encoder uses a pyramidal resize module and inception network to leverage the priors at different resolution scales to improve the efficiency of the compressed latents. The decoder utilizes a residual network to synthesize the images with more nonlinearity. The separate entropy model is adopted to better predict the prior probability model of the latent representation. The final experiment results show that our approach yields a state-of-the-art image compression system.*

## 1. Introduction

Lossy image compression based on Convolutional Neural Networks (CNNs) has become an active area of research in recent years. Many works have revealed great potentials in learned image compression [8, 6, 4, 2]. Most of the achievements are based on an autoencoder structure, which consists of an encoder, mapping the input image pixels to a latent code space by generating a compact representation, and a decoder, an approximate inverse function that reconstructs images close to input according to the latents.

The autoencoder based lossy image compression aims at representing the images in as little bits as possible but with as good quality as possible, which results in so called rate-distortion trade-off. In order to keep good balance between the bitrate and distortion, people often use a two-fold method. The first is to find a most approximate entropy model for the latent representation for optimizing the length of bitstream. The second is to get a more effective latent representation for reconstructing the image precisely.

Currently, one of the widely studied end-to-end image compression frameworks is proposed by Ballé et al. [3]. They use a noise-based relaxation to replace the round quantization function, which can apply to the gradient descent methods. They also introduce a Gaussian Scale Mixture (GSM) [9] model where the scale parameters are conditioned on a hyperprior to model the latents' probability. Minnen et al. [5] improve the GSM-based entropy model by generalizing the hierarchical GSM model to a Gaussian mixture model and adding an autoregressive component. It is the first learning-based method outperforming BPG on both PSNR and MS-SSIM distortion metrics.

Our architecture is inspired by the successful work of [3] and [5]. We extend the encoder with a pyramidal resizing module and an inception function to sufficiently extract the features of input image, and use a residual decoder to reconstruct the input image accurately. We also model the prior probability of the compressed representation precisely with a separate context model and entropy parameter model.

## 2. Framework

The proposed framework is derived from the autoencoder architecture, as shown in Figure 1. It consists two autoencoder networks. The first one compresses the input image $x$ into a latent representation $y$, and quantizes it to a discretely valued vector $\hat{y}$. It is then coded to a bitstream through an arithmetic encoder according to the predicted probability. Then, the decompressed latents $\hat{y}$ is transformed back to get the reconstructed image $\tilde{x}$ using a decoder transform. The second autoencoder is a hyperprior network. It mainly focuses on capturing the spatial dependencies in the latent representation to model the distribution of latents accurately with the *Context Model* and *Entropy Parameter* model jointly.

*wensihan@cn.fujitsu.com

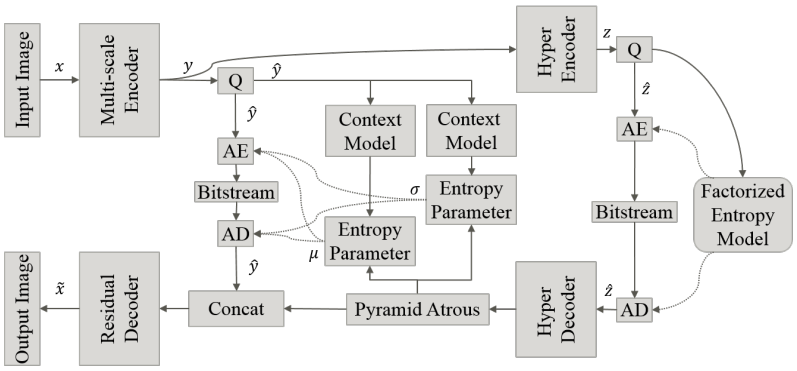


Figure 1. Framework of image compression. *Q*: quantization, replaced with an additive uniform noise when training. *AE*: arithmetic encoder. *AD*: arithmetic decoder.

## 2.1. Multi-scale Encoder

The *Multi-scale Encoder* consists of a pyramidal resize model, as shown in Figure 2. It extracts the image feature across 4 scales by resizing the input image to different sizes with a bilinear interpolation. The resized images are then aligned and merged to discover joint structure across different scales through a convolution layer. Generally speaking, when we use the convolutional neural network to extract the coefficient maps from image, the global and coarse information is exploited from the deeper layers, whereas local and fine information is presented from the shallower layers. Therefore, we use a network of four layers to get the global and high-level information from the original image, and employ a network of one layer to get the detailed features for the other resized images.

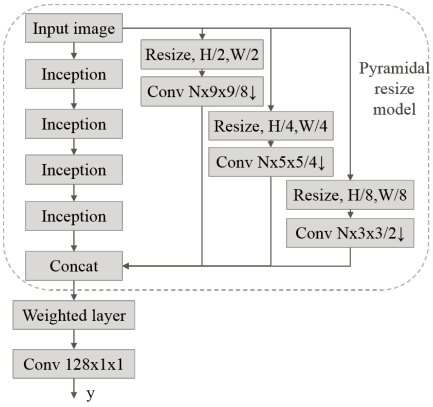


Figure 2. Multi-scale Encoder. The dashed box indicates the pyramidal resize model. Conv: convolution layer. N×3×3/2: Number of feature×kernel height×kernel width/sampling stride. While ↓ means downsampling and ↑ indicates upsampling. N is set to 192.

The inception network [7] is well known to exploit multi-scale features by using different kernels. We adopt an inception module for the original image. To better fit for our network, we use the same channel number for different kernels and concatenate them together. Then a 1x1 convolution layer is used to decide which kernel is the most important and get the fused output, as shown in Figure 3.

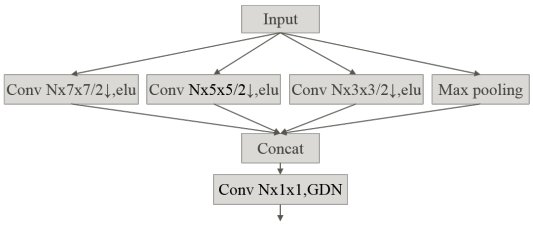


Figure 3. Inception network. GDN: generalized divisive normalization [1].

## 2.2. Residual Decoder

*Residual Decoder* is designed as shown in Figure 4. A two-layers residual network is added between the adjacent convolution layers to increase the nonlinearity of the decoder, which benefits for enhancing the quality of reconstructed image. Moreover, some multi-scale assistant information, generated by processing the hyperprior's output through a *Pyramid Atrous* model, is concatenated with the quantized latent representation to get more features feeding for the decoder network.

## 2.3. Entropy Model

The entropy model is used for learning a probabilistic model over quantized latents used for entropy coding. We use a conditional Gaussian Mixture Model (GMM) [5] in this work. The parameter of GMM is generated by an *Entropy Parameter* model by processing the information from the *Context Model* and the hyperprior network. With the predicted parameters of $\mu$ and $\sigma$, the discrete representa-

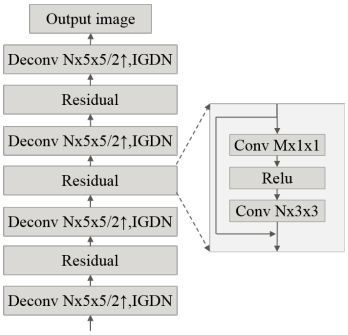


Figure 4. Residual Decoder. Deconv: upsampled convolution. IGDN: inverse GDN [1]. M is set to 128.

tion's probability is calculated with Equation 1.

$$p(\hat{y}|\hat{z}) = \prod_i (\mathcal{N}(\mu_i, \sigma_i^2) * \mu(-\frac{1}{2}, \frac{1}{2}))(\hat{y}_i) \tag{1}$$

For the hyper-latents $\hat{z}$, a non-parametric, fully factorized density model is used as [3] to predict the probability of side information. Then, the entropy of generated bitstream containing the compressed latents $R_{\hat{y}}$ and hyper-latents $R_{\hat{z}}$ can be defined as Equation 2.

$$R = \underbrace{\sum(-log_2(p(\hat{y}|\hat{z}))}_{R_{\hat{y}}} + \underbrace{\sum -log_2(p(\hat{z}))}_{R_{\hat{z}}} \tag{2}$$

However, generating the parameters of $\mu$ and $\sigma$ through one single *Context Model* and *Entropy Parameter* model has some limitations. There may be some correlation between $\mu$ and $\sigma$, but the independence between them should not be ignored. Therefore, in order to get a more accurate distribution of the latents, we use a *Context Model* and *Entropy Parameter* model for $\mu$ and $\sigma$ respectively. Details about the individual network layers are outlined in Table 1.

| Context Model($\mu$&$\sigma$) | Entropy Parameter($\mu$&$\sigma$) |
|---|---|
| Masked 128×3×3 | Conv 384×1×1 |
| Masked 128×3×3 | Conv 192×1×1 |
| Masked 128×3×3 | Conv 128×1×1 |

Table 1. Parameters of Context Model and Entropy Parameter. "Masked" corresponds to masked convolution as in [4].

With the independence of networks, an absolute function is added before the entropy model for $\sigma$. For the value of $\sigma$ mainly stands for the variance of data, and a positive input will provide a better representation. Furthermore, we use three layers instead of one layer for the *Context Model*. With this modification, the receptive field of the convolution layer increase from 3x3 to 7x7 which contains more information from the quantized latents to better predict the latents' probability. The hyperprior is similar to [3] with the channel number increase to 192.

### 2.4. Weighted layer

After analyzing the time consuming of the whole network, we find that the autoregressive network takes most of the time. It needs to code the latents pixel by pixel, and takes a lot of time if the size of bottleneck becomes large.

To solve this problem, one intuitional method is to reduce the bottlenecks by decreasing the number of feature maps. However, if we cut the number of feature maps directly through a convolution layer, the quality of reconstructed image will turn worse correspondingly. Therefore, we propose to add a weighted layer before the convolution layer. It is used to provide a weight to different channels to selectively enhance the useful features and surpass useless ones. With this selection, we can maintain more important information when facing to dimensionality reduction.

The structure of weighted layer is shown in Figure 5. We first use a global average pooling layer to generate channel-wise statistics. Then two convolution layers are used to better learning the nonlinear interaction between channels.

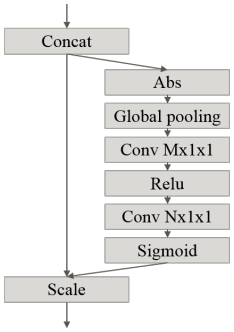


Figure 5. Structure of weighted layer.

## 3. Experiment

We use the dataset from Challenge on Learned Image Compression (CLIC), and extra collect about 5000 high quality images to maintain the diversity of training dataset, making the total size of the dataset around 7000. We use two kinds of common distortion measures, Mean Square Error (MSE) and perceptual metric MS-SSIM to train the network with the loss function:

$$Loss = R + \lambda \times D \tag{3}$$

where $D$ is the distortion measured as $\|x - \tilde{x}\|^2$ for MSE or MS-SSIM defined in [10], $R$ is the entropy of latents $\hat{y}$ and $\hat{z}$. $\lambda$ controls the tradeoff between rate and distortion.

To demonstrate the effect of our models intuitively, we analyze the improvement of different sections trained with MSE. The "Original" means the network we implement based on Minnen's [5] work. The "Separate Entropy", "Multi-scale Encoder" and "Residual Decoder" means the network added with the named portion step by step.

| methods | bitrate | PSNR | MS-SSIM |
|---|---|---|---|
| Original | 0.149 | 31.03 | 0.956 |
| Separate Entropy | 0.149 | 31.29 | 0.958 |
| Multi-scale Encoder | 0.149 | 31.69 | 0.96 |
| Residual Decoder | 0.149 | 31.72 | 0.96 |

Table 2. Performance of the different models.

From Table 2, we can observe that the added network can exactly enhance the compression quality with a considerable improvement, especially for the *Multi-scale Encoder*. However, comparing to the "Original", the complexity of proposed network added with the whole portions enhances correspondingly. Table 3 shows the increase of time.

| methods | Encoder (%) | Decoder (%) |
|---|---|---|
| Original | 100 | 100 |
| Proposed | 145 | 115 |

Table 3. Comparison of computational complexity.

| methods | bitrate | PSNR | MS-SSIM |
|---|---|---|---|
| BPG | 0.149 | 30.84 | 0.948 |
| Proposed + MSE | 0.149 | 31.72 | 0.96 |
| Proposed + MS-SSIM | 0.149 | 29.60 | 0.974 |

Table 4. Evaluation results of CLIC 2019 on validation dataset.

We provide a comparison of our approaches and BPG on validation dataset under the CLIC requirements. From Table 4, we can see that when trained with MSE, the proposed approach can surpass the BPG in both PSNR and MS-SSIM, outperforming the traditional method. When the model is trained with MS-SSIM, the perceptual score is significantly enhanced (0.948 $\rightarrow$ 0.974).

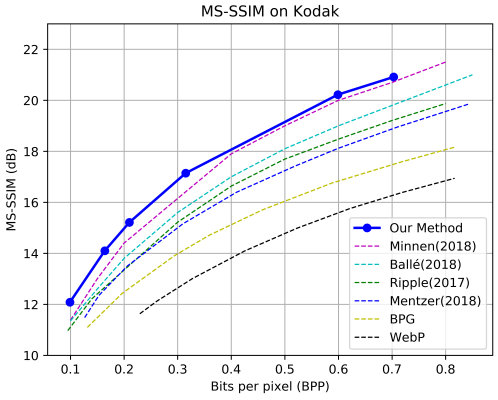


Figure 6. Evaluation on the Kodak image dataset using MS-SSIM.

To make a more direct comparison, Figure 6 shows RD curves for Kodak using MS-SSIM. The score of MS-SSIM is formulated as $-10log_{10}(1 - MS\text{-}SSIM)$. We can see our method achieves a state-of-the-art performance.

Moreover, we use eight models to compress the images separately, and select the most appropriate model for every image. The average bitrate of those models range from 0.131 to 0.172. Table 5 shows a comparison of one fixed model and eight selected models. We can see that, with the selected models, we can make a better allocation and constraint of the bits to match the demand of CLIC.

| methods | bitrate | PSNR | MS-SSIM |
|---|---|---|---|
| Fixed model | 0.146 | 28.32 | 0.970 |
| Selected models (Hyper) | 0.150 | 28.36 | 0.972 |

Table 5. Evaluation results of CLIC 2019 on test dataset.

## 4. Conclusion

In this study, we present a novel autoencoder for low bit-rate image compression. With the multi-scale encoder, the features of input image is fully extracted. The obtained latent representation can be reconstructed by residual decoder. The separate entropy model provides a closer prediction of the prior probability for latent representation. Our experiments compared with other methods shows that this approach has achieved a state-of-the-art performance in image compression. In future work, a more effective compression network with higher evaluation scores and less computational load will be exploited.

## References


[1] Johannes Ballé, Valero Laparra, and Eero P Simoncelli. Density modeling of images using a generalized normalization transformation. *arXiv preprint arXiv:1511.06281*, 2015.

[2] Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704*, 2016.

[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.

[4] 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.

[5] 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.

[6] 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.

[7] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[8] 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.

[9] Martin J Wainwright and Eero P Simoncelli. Scale mixtures of gaussians and the statistics of natural images. In *Advances in neural information processing systems*, pages 855–861, 2000.

[10] 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.