Neural Image Compression with Quantization Rectifier

Anonymous Authors[1]

Abstract

Neural image compression has been shown to outperform traditional image codecs in terms of rate-distortion performance. However, quantization introduces errors in the compression process, which can degrade the quality of the compressed image. Existing approaches address the train-test mismatch problem incurred during quantization, the random impact of quantization on the expressiveness of image features is still unsolved. This paper presents a novel quantization rectifier (QR) method for image compression that leverages image feature correlation to mitigate the impact of quantization. Our method designs a neural network architecture that predicts unquantized features from the quantized ones, preserving feature expressiveness for better image reconstruction quality. We develop a soft-to-predictive training technique to integrate QR into existing neural image codecs. In evaluation, we integrate QR into state-of-the-art neural image codecs and compare enhanced models and baselines on the widely-used Kodak benchmark. The results show consistent coding efficiency improvement by QR with a negligible increase in the running time.

1. Introduction

Neural network (NN)-based image compression methods (Ballé et al., 2016; 2018; Cheng et al., 2020; Minnen et al., 2018) have shown superior coding efficiency to those of the conventional compression methods, such as BPG (Bellard, 2018) and JPEG2000 (Joint Photographic Experts Group, 2000). Quantization discretizes image features by mapping continuous values to a limited set of discrete values for entropy coding, compressing the image (Huffman, 1952; Witten et al., 1987). While current quantization methods address train-test mismatch, the random effects on feature expressiveness remain unresolved. Quantization uniformly maps continuous values to a single discrete value, introducing different degrees of noise depending on feature variability. It also unpredictably alters feature expressiveness. For instance, the quantization of features from the range of $[-0.5, 0.5)$ to zero introduces noises in the range of $(-0.5, 0.5]$. More importantly, quantization alters the expressiveness of the latent features in an unpredictable way. In this paper, we propose a novel quantization rectifier (QR) that leverages spatial correlation in images to mitigate the impact of quantization. Specifically, we design a neural network architecture that predicts unquantized features from the quantized ones. To seamlessly integrate QR into a neural image codec, we introduce a soft-to-predictive (STP) training method. Here, we first softly train the original image compression model end-to-end until convergence. Then, we freeze the encoder network with hard quantization and optimize the decoder network, along with the QR network. QR bridges the gap between original and quantized features, preserving feature expressiveness for improved image reconstruction quality. For evaluation, we incorporate our method into state-of-the-art neural network-based compression methods (Ballé et al., 2016; 2018; Cheng et al., 2020; Minnen et al., 2018). We consistently improve all baseline models by up to 0.21 dB (PSNR) and 0.25 dB (MS-SSIM) without affecting the bitrate. QR is lightweight, with a minimal increase (0.7-5.4%) in running time for most baselines.The contributions of this study are summarized as follows:

- We propose QR, a method that corrects quantized image features through prediction, preserving feature expressiveness and improving coding efficiency.
- We develop the STP training procedure and a hyperparameter exploration algorithm, enabling seamless integration of QR with existing neural image codecs.
- We extensively evaluate QR on state-of-the-art neural image codecs, which demonstrate the superiority of QR consistently.

2. Related Works

Quantization plays a vital role in image compression, enabling efficient storage and entropy coding (Huffman, 1952;

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

Witten et al., 1987). Recent advancements, including Additive Uniform Noises (Ballé et al., 2016; 2018) and Straight-Through Estimator (Mentzer et al., 2018; Theis et al., 2017; Yin et al., 2019) aim to tackle the train-test discrepancy arising from quantization. Soft-to-hard annealing (SA) approximates quantization using a differentiable function resembling hard rounding, but its training is fragile, requiring empirical determination of the annealing function. The Soft-Then-Hard (STH) strategy (Guo et al., 2021) first learns a soft latent space and then resolves the train-test mismatch with hard quantization, partially addressing the issue. While these approaches address the train-test discrepancy, the expressiveness of latent features is still unpredictably affected by quantization. Our proposed approach effectively mitigates the impact of quantization on feature expressiveness and can be easily integrated into these techniques.

3. Proposed Method

3.1. Formulation of Learned Compression Models

According to recent works (Ballé et al., 2018; Cheng et al., 2020; Gao et al., 2022), the general procedure of neural image compression can be formulated as follows:

$$y = g_a(x; \phi) \tag{1}$$
$$\hat{y} = Q(y) \tag{2}$$
$$\hat{x} = g_s(\hat{y}; \theta), \tag{3}$$

where x, $\hat{x}$, y, and $\hat{y}$ are the raw image, the reconstructed image, the latent feature before quantization, and the quantized latent feature, respectively. ϕ and θ are parameters of the encoder and decoder. In the encoding process g_a, latent feature y is produced from the raw image x (Eq. 1). For the quantization step Q, latent feature y is quantized (rounded) to $\hat{y}$ (Eq. 2). During the decoding process g_s, quantized $\hat{y}$ is fed into the decoder network to obtain the reconstructed image $\hat{x}$ (Eq. 3). Since the quantization operation Q is not differentiable in training, the quantization Q is typically approximated by adding a uniform noise $\mathcal{U}(-0.5, 0.5)$ to the input. We define a probability model $p(\hat{y}; \phi)$, which is parameterized by ϕ to compute the probability mass function (PMF) of quantized feature $\hat{y}$ as shown in Eq. 4.

$$\begin{aligned} p(\hat{y}; \phi) &= \prod_i \int_{\hat{y}^i - 0.5}^{\hat{y}^i + 0.5} p(\hat{y}^i; \phi) d\hat{y}^i \\ &= \prod_i \left(F\left(\hat{y}^i + 0.5; \phi\right) - F\left(\hat{y}^i - 0.5; \phi\right) \right), \end{aligned} \tag{4}$$

where $F(\cdot; \phi)$ is the cumulative distribution function of $p(\cdot; \phi)$ and i iterates over all symbols in $\hat{y}$. The goal of the image compression task is to minimize the rate-distortion loss function as shown in Eq. 5 with respect to parameters θ and ϕ.

$$\mathcal{L}_{\theta,\phi} = \mathcal{R}(\hat{y}) + \lambda \mathcal{D}(x, \hat{x}) \tag{5}$$
$$= \underbrace{\mathbb{E}\left[-\log_2 p(\hat{y}; \phi)\right]}_{\text{rate}} + \underbrace{\mathcal{D}(x, \hat{x})}_{\text{distortion}}, \tag{6}$$

where the number of bits required to encode quantized $\hat{y}$ is represented by $\mathcal{R} = \mathcal{R}(\hat{y})$. The distortion between reconstruction image $\hat{x}$ and the original image x is calculated by $\mathcal{D}(x, \hat{x})$, which is commonly evaluated by the peak signal-to-noise ratio (PSNR) or multiscale structural similarity (MS-SSIM) (Wang et al., 2003) of the raw and reconstructed images. The encoder and entropy model parameter ϕ, and the decoder parameter θ are jointly optimized to reduce the rate-distortion cost $\mathcal{R} + \lambda\mathcal{D}$, where λ controls the rate-distortion trade-off.

The hyperprior adopted in many existing works (Ballé et al., 2018; Mentzer et al., 2018) is omitted from the above formulation for simplicity. However, such simplification does not affect the generality of our approach.

3.2. Quantization Rectifier

Definition. To mitigate the random impact of quantization, we introduce the quantization rectifier (QR). We take y from Eq. 1 as the input and train the corrected feature through a QR network, which is used to predict unquantized features from quantized ones. The effectiveness of the QR relies on its network design. The insight is to exploit the spatial correlation within the image feature that recovers itself even under noise. Inspired by the success of Model Diffusion (Ho et al., 2020) in image denoising, we design the QR network as shown in Fig. 1. The QR network stays between the quantization step and the decoder. It consists of convolutional layers (conv), residual blocks (res-block), and attention layers (attn) that spatially correlate quantized features $\hat{y}$. We then add the output of the last conv to the quantized feature $\hat{y}$ and acquire the corrected feature $\tilde{y}$ (Eq. 7). Next, $\tilde{y}$ replaces $\hat{y}$ in the decoding phase as shown in Eq. 8.

$$\tilde{y} = \text{QR}(\hat{y}) \tag{7}$$
$$\hat{x} = g_s(\tilde{y}; \theta). \tag{8}$$

A more detailed architecture of the QR is described in Appendix A. Compared to the network in Model Diffusion (Ho et al., 2020), our network is configured with fewer layers for efficiency while being effective. The QR is a versatile module, which can be seamlessly integrated into any neural image compression method that can be broken down into an encoder, a quantization module, and a decoder. There is no need to make significant modifications to the encoder and decoder components of the original image compression model.

Learning Rectifier. To facilitate the learning of the QR, we replace the loss function in Eq. 5 with Eq. 9 that adds a

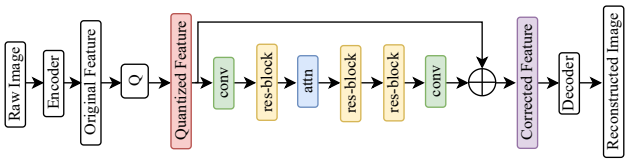


Figure 1. Quantization Rectifier Architecture.

feature distance term to the original formulation.

$$\mathcal{L}_{\theta,\phi,\psi} = \underbrace{\mathcal{R}(\hat{y})}_{\text{rate}} + \lambda \underbrace{\mathcal{D}(x,\hat{x})}_{\text{distortion}} + \alpha \underbrace{\mathcal{D}^f(y,\tilde{y})}_{\text{feature distance}}, \quad (9)$$

where α is the learning coefficient controlling the relative learning rate of QR compared to that of the rate and distortion. An algorithm describing the exploration of α will be further detailed in Sec. 3.3. $\mathcal{D}^f(y,\tilde{y})$ is a measure of the distance between the original and quantized image features, i.e., feature distance. The design of the feature distance is crucial to the learning of QR, where a smaller distance should reflect better preservation of feature expressiveness. We consider four commonly used distance terms: L1 distance (Lasserre, 2009), L2 distance (Park & Koo, 1990), Smooth L1 distance (Girshick, 2015), and cosine similarity (Salton & McGill, 1986; Deza & Fernández, 2009; Manning et al., 2008; Ramos, 2003). Our empirical study shows that minimizing the L2 distance yields the best image quality with QR. Other measures like the L1 distance do not provide as promising results. Hence, we formulate the feature distance as in Eq. 10.

$$\mathcal{D}^f(y,\tilde{y}) = \|\mathbf{y} - \tilde{\mathbf{y}}\|_2. \quad (10)$$

3.3. Soft-to-predictive Training

Training is an essential step that integrates the QR into the neural image codec. We find the straightforward end-to-end training is sub-optimal due to the inter-dependency of the training of the codec and the QR. First, the learning of the QR relies on the stability of its input, which is the latent feature. Second, the stability of the latent features is contingent upon the convergence of the prediction network. Even a slight perturbation in the latent feature would disrupt the training process of the prediction network. Consequently, the disturbed prediction network would further affect the stability of the latent features. In such a vicious cycle, where the latent feature and prediction network constantly fail to converge, the overall training process is sub-optimal.

To address the sub-optimal training issue, we develop a soft-to-predictive (STP) training technique consisting of the soft and predictive training phases. In the soft training phase, the image is reconstructed based on Eq. 1, Eq. 2, and Eq. 3. Meanwhile, we learn parameters of the encoder (θ), the decoder (ϕ), and QR (ψ) based on the loss function Eq. 9 softly with additive uniform noise. Although we do not apply Eq. 7 and Eq. 8 in the generation of $\hat{x}$, QR will still be learned to predict the feature, which warms up the next phase. In the predictive training phase, the image is reconstructed based on Eq. 1, Eq. 2, Eq. 7, and Eq. 8. The encoder θ is fixed with its output being hard quantized while the decoder (ϕ) and QR (ψ) are optimized according to the loss function in Eq. 11.

$$\mathcal{L}_{\theta,\psi} = \mathcal{D}(x,\hat{x}) + \alpha\mathcal{D}^f(y,\tilde{y}), \quad (11)$$

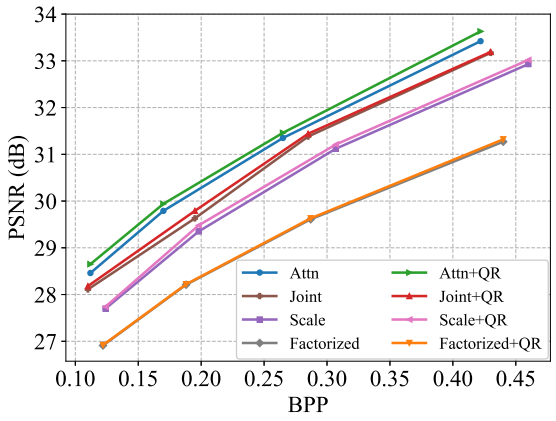
where only θ and ψ are optimized. The bitrate $\mathcal{R}$ is omitted in Eq. 11 as its parameters are no longer optimized. λ is also dropped in Eq. 11 for simplicity. In the predictive training phase, the latent feature and bitrate stay fixed, which stabilizes the training of QR.

During the predictive training phase, choosing the rectifier learning coefficient α in Eq. 9 is non-trivial as its optimal value varies across different models and compression quality. Appendix B demonstrate the learning coefficient exploration algorithm we proposed and its results.

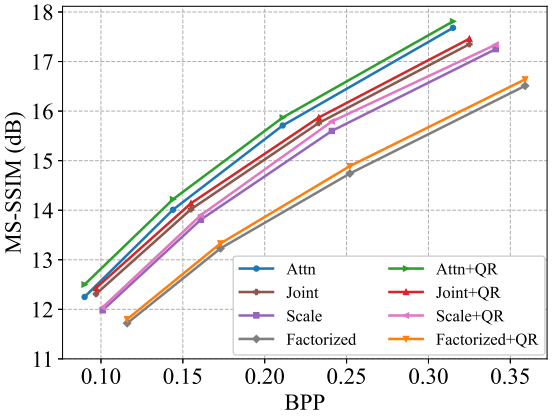
4. Experiments

4.1. Experimental Setup

To demonstrate the effectiveness of the QR, we apply it to four baseline neural image compression models: Factorized Prior (Ballé et al., 2016), Scale Hyperprior (Ballé et al., 2018), Joint Hyperprior (Minnen et al., 2018), and Attention-based Joint Hyperprior (Cheng et al., 2020), denoted by "Factorized", "Scale", "Joint", and "Attn", respectively. The selected baseline models capture dominant neural image compression architectures. Accordingly, the models enhanced by the QR are represented by "Factorized+QR", "Scale+QR", "Joint+QR", and "Attn+QR", respectively. According to CompressAI (Gravano et al., 2021), the four baseline models are previously trained on 256×256 image patches randomly extracted and cropped from the Vimeo90K dataset (Xue et al., 2019) with a batch size of 32 using the Adam optimizer (Kingma & Ba, 2014). The test is performed on the commonly used Kodak image dataset (Hersch, 2001). We compare the performance of the four enhanced models against the corresponding baseline models in terms of rate and distortion trade-offs. Following many existing works (Ballé et al., 2016; 2018), the rate is measured by bits per pixel (bpp) while the distortion is measured by either PSNR or MS-SSIM. MS-SSIM is converted to decibels ($-10\log_{10}(1-\text{MS-SSIM})$) to illustrate the difference clearly. For fairness, both baseline and enhanced models are optimized with MSE or MS-SSIM, depending on the distortion metric (PSNR or MS-SSIM). An description of the training configuration and a more detailed illustration of metrics and is in Appendix C.



(a) PSNR



(b) MS-SSIM

Figure 2. Coding efficiency of baseline models and their enhanced versions by QR in terms of PSNR and MS-SSIM.

4.2. Coding Efficiency Improvement

Fig. 2 compares the coding efficiency of the baseline models without and with the proposed QR. Every point on curves in Fig. 2 represents the bpp and distortion (PSNR or MS-SSIM) averaged over the Kodak image dataset (Hersch, 2001) different compression quality levels $q \in \{1, 2, 3, 4\}$. For a specific baseline model at any given quality level, the average bpp value remains the same after applying QR. The reason is that the soft training process for the encoder with QR is identical to the training of the baseline model. After soft training, the encoder is fixed, so the average bpp value would not change. Comparing the baseline models to their corresponding enhanced versions in Fig. 2, we notice QR consistently improves all baseline models at various compression qualities in terms of both PSNR and MS-SSIM. Further, for a relatively more complex model, e.g., Attn, QR shows improvement by a wider margin than that of a simple model like Factorized. We speculate that a more complex model, with more parameters, can better leverage the reduced effect of quantization towards better image quality. Moreover, the improvement by QR is more evident utilizing MS-SSIM than utilizing PSNR.

The image quality of all models utilizing our proposed QR method surpasses that of the baseline models in both PSNR and MS-SSIM. Among all enhanced models, Attn+QR demonstrates the most substantial enhancement in PSNR, with an average quality improvement of 0.17 dB and a maximum improvement of 0.21 dB. With MS-SSIM, Attn+QR is still the best-performing one with a 0.19 dB average and 0.25 dB maximum improvement over Attn. While Factorized+QR exhibits a relatively smaller improvement compared to the other enhanced models in PSNR, its improvement is significant in MS-SSIM, with an average of 0.12 dB and a maximum of 0.15 dB. The numerical results are summarized in Tab. 3 shown in Appendix D.

A detailed evaluations regarding quantization error reduction of the proposed QR component can be found in Appendix E.

4.3. Processing Speed

Table 1. Runtime cost increase in milliseconds after applying the quantization rectifier, evaluated on NVIDIA RTX 2080 Ti GPU (2080) and NVIDIA RTX 3090 Ti GPU (3090).

HW	Attn	Joint	Scale	Factorized
2080	5.2%	0.7%	4.6%	16.6%
3090	5.4%	0.8%	5.4%	17.1%

In Tab. 1, we compare the average processing time per frame of the baseline models and their enhanced versions by QR, including encoding and decoding, on the Kodak dataset. During the time measurement, we factor out the time spent in the conversion between symbol likelihoods and bits to precisely show the impact of QR on the neural network-related computation. Tests are performed on NVIDIA RTX 2080 Ti and NVIDIA RTX 3090 Ti GPUs. Our method slightly increases processing time by 0.7-5.4% for most baselines (Attn, Joint, and Scale), while Factorized is more affected as its processing time is already short due to its simplest network architecture.

5. Conclusion

We introduce a Quantization Rectifier (QR) method to enhance neural image compression. QR utilizes spatial correlation in images to predict features before quantization, preserving their expressiveness. Our method includes a soft-to-predictive training approach that allows seamless and optimal integration of QR into existing neural image codecs. Experimental results consistently demonstrate the effectiveness of QR across various state-of-the-art neural image codecs.

References

Ballé, J., Laparra, V., and Simoncelli, E. P. End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704*, 2016.

Ballé, J., Minnen, D., Singh, S., Hwang, S. J., and Johnston, N. Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436*, 2018.

Bellard, F. Bpg image format, 2018. URL https://bellard.org/bpg/.

Cheng, Z., Sun, H., Takeuchi, M., and Katto, J. Learned image compression with discretized gaussian mixture likelihoods and attention modules. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Deza, E. and Fernández, M. A. Cosine similarity measures for efficient document retrieval. *Information Processing & Management*, 45(1):67–80, 2009.

Flickr. Flickr image and video dataset. https://www.flickr.com/services/api/flickr.photos.search.html, 2021. Accessed on May 10, 2023.

Gao, C., Xu, T., He, D., Qin, H., and Wang, Y. Flexible neural image compression via code editing, 2022.

Girshick, R. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

Gravano, A., Hoyet, M., Liutkus, A., and Daudet, L. CompressAI: A pytorch library for efficient and composable deep learning-based compression of images and neural networks. https://github.com/InterDigitalInc/CompressAI, 2021. Accessed: May 14, 2023.

Guo, Z., Zhang, Z., Feng, R., and Chen, Z. Soft then hard: Rethinking the quantization in neural image compression. In *International Conference on Machine Learning*, pp. 3920–3929. PMLR, 2021.

Hersch, R. D. Digital color image rendering using scene-adaptive color remapping. In *Proceedings of IST/SPIE Electronic Imaging*, volume 4299, pp. 34–44. International Society for Optics and Photonics, 2001. Kodak Lossless True Color Image Suite used in experiments.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Huffman, D. A. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

Joint Photographic Experts Group. Jpeg 2000. Part 1: Core coding system. ITU-T Recommendation T.800 | ISO/IEC 15444-1, 2000. Available online at https://www.itu.int/rec/T-REC-T.800/en and https://www.iso.org/standard/28902.html.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Lasserre, V. Least absolute deviation estimation for arma models based on l1-norm innovations. *Signal Processing*, 89(7):1359–1369, 2009.

Manning, C. D., Raghavan, P., and Schütze, H. Introduction to information retrieval. In *Proceedings of the 1st annual ACM SIGIR conference on Research and development in information retrieval*, pp. 333–334. ACM, 2008.

Mentzer, F., Agustsson, E., Tschannen, M., Timofte, R., and Van Gool, L. Conditional probability models for deep image compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4394–4402, 2018.

Minnen, D., Ballé, J., and Toderici, G. Joint autoregressive and hierarchical priors for learned image compression. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pp. 10794–10803, 2018.

Park, P. and Koo, J.-Y. Fast and stable signal recovery. *SIAM Journal on Scientific and Statistical Computing*, 11(2): 273–294, 1990.

Ramos, J. Using tf-idf to determine word relevance in document queries. *In Proceedings of the First Instructional Conference on Machine Learning*, 242:29–48, 2003.

Salton, G. and McGill, M. J. Introduction to modern information retrieval. *Journal of the American Society for Information Science*, 37(3):173–174, 1986.

Theis, L., Shi, W., Cunningham, A., and Huszár, F. Lossy image compression with compressive autoencoders. *arXiv preprint arXiv:1703.00395*, 2017.

Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. Multiscale structural similarity for image quality assessment. *IEEE Transactions on Image Processing*, 13 (4):600–612, 2003.

Witten, I. H., Neal, R. M., and Cleary, J. G. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, 1987.

Xue, T., Chen, B., Wu, J., Wei, D., and Freeman, W. T. Video enhancement with task-oriented flow. *International Journal of Computer Vision (IJCV)*, 127(8):1106–1125, 2019.

Yin, P., Lyu, J., Zhang, S., Osher, S., Qi, Y., and Xin, J. Understanding straight-through estimator in training activation quantized neural nets. *arXiv preprint arXiv:1903.05662*, 2019.

A. Detailed Quantization Rectifier Architecture

In accordance with Sec. 3.2, the architecture presented in Fig. 3 provides a comprehensive illustration of the Quantization Rectifier (QR) network. For the purpose of clarity, the encoder and decoder components depicted in Fig. 1 are omitted in this particular representation. The architecture consists of convolutional layers (conv), which are intertwined with groups of residual blocks (res-block) and multi-head attention layer (attn). These attention layers spatially correlate the quantized features $\hat{y}$.

Initially, the quantized feature with a dimension of 192 is inputted into a conv with a dimension of 512, employing a kernel size of 7×7 and a padding of 3. The output of this conv is then directed to the first set of grouped res-block, consisting of eight groups, with each group having a dimension of 64 and a kernel size of 3×3. The resulting output from the res-block is normalized through layer normalization (layer-norm) and serves as the input for a multi-head attn with four heads, where each head possesses a dimension of 32. The output of this attn is subsequently added to the output of the first res-block layer. This summation then serves as the input for the second set of res-block groups, which mirror the architecture of the first group. The output of this second set of res-block is concatenated with the output of the initial conv and is then fed into the final group of res-block. The concatenated output undergoes a final conv with a dimension of 192 and a kernel size of 1×1. Lastly, the output from this conv is added to the quantized feature $\hat{y}$, resulting in the corrected feature $\tilde{y}$, which retains the same dimension as the original quantized feature $\hat{y}$.

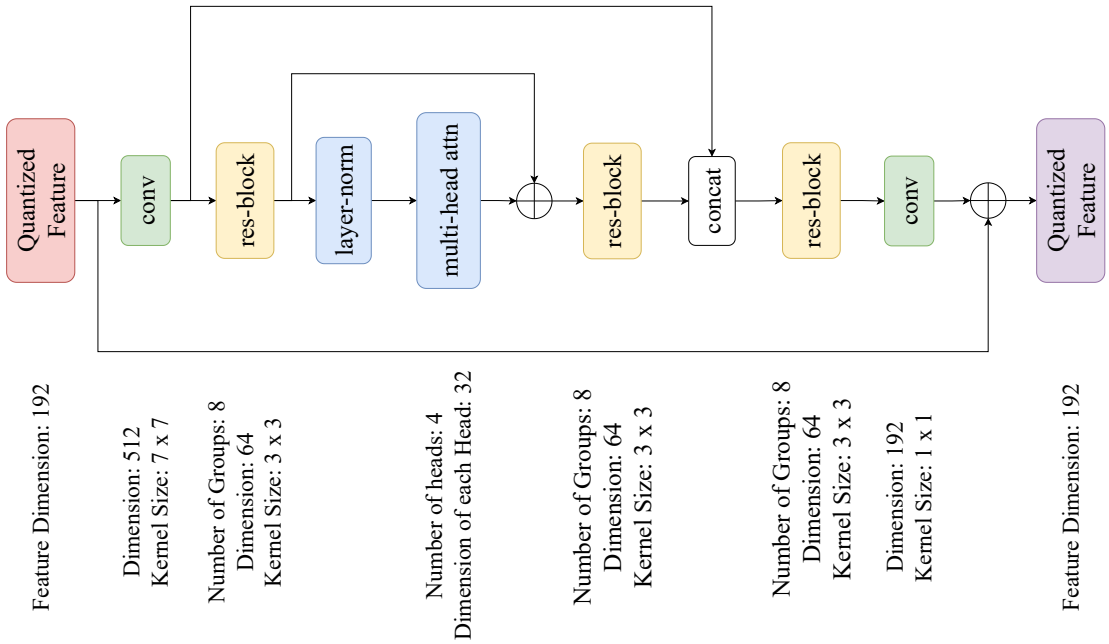


Figure 3. Detailed Architecture of the Quantization Rectifier.

B. Learning Coeeficient Exploration in Training

Rectifier Learning Coefficient Exploration Algorithm. During the predictive training phase as mentioned in Sec. 3.3, choosing the rectifier learning coefficient α in Eq. 9 is non-trivial as its optimal value varies across different models and compression quality. Fig. 4 compares the image quality tested on the Kodak dataset, resulting from different rectifier learning coefficients. An optimal coefficient, e.g., 10^{-3}, allows the image quality to start at a high value and converge in a few epochs, e.g., 6 epochs in Fig. 4. If the learning coefficient is too small, e.g., 10^{-6}, it will take a long time for the model to converge at a sufficiently good image quality. Conversely, if the learning coefficient is too large, e.g., 10^{2}, the rectifier changes too fast for the decoder to converge, which degrades image quality. To tackle this issue, we introduce a rectifier learning coefficient exploration method that automatically finds the optimal learning coefficient for different models and compression quality.

One of our key findings is, there exists an optimal learning coefficient where increasing or decreasing it would only monotonically degrade coding efficiency. Based on this finding, we describe the exploration strategy for a specific model

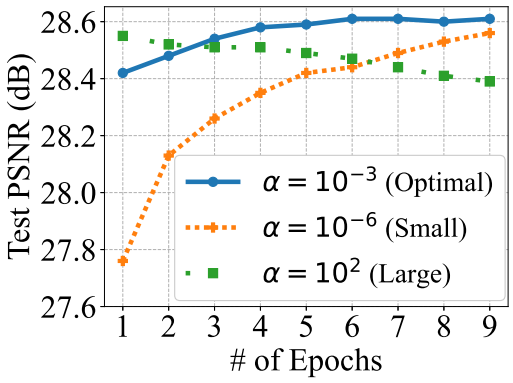


Figure 4. The impact of rectifier learning coefficient on image quality.

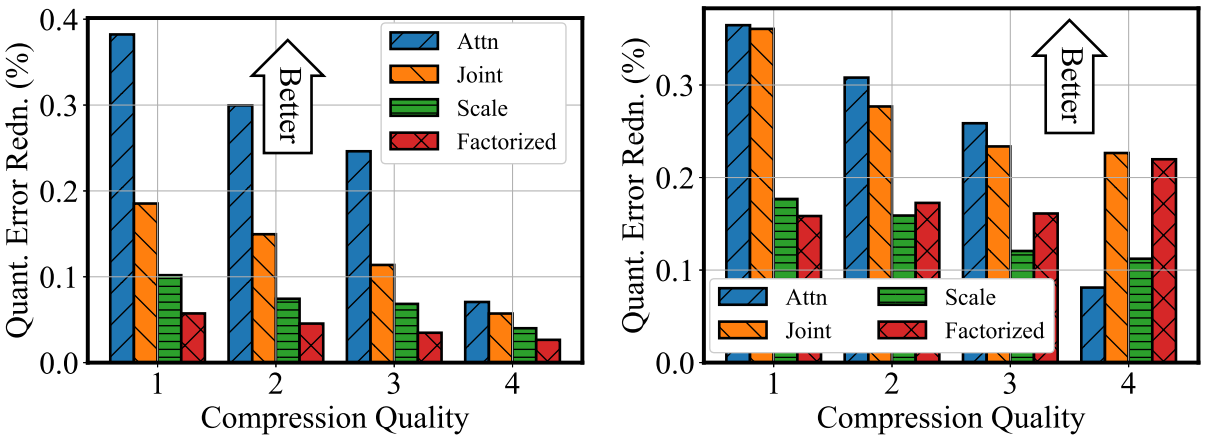
and compression quality as follows: i) start the exploration at an initial learning coefficient $\alpha = \alpha_{max}$, ii) train the codec as specified in Sec. 3.3 using α until the loss (Eq. 9) stops improving for M consecutive epochs, iii) multiply the learning coefficient α by 0.1, iv) continue step ii) if the learning coefficient is no smaller than a pre-defined lowest learning coefficient α_{min} or stop exploration otherwise. M is set to 3, which confidently finds the non-improving loss. Considering the efficiency of exploration, we adopt a relatively small dataset in exploration, which remains as effective as the big one in predictive training.

Table 2. Rectifier Learning Coefficient Exploration: distortion measured in PSNR (dB) at compression qualities $q \in 1, 2, 3, 4$ using Flickr image dataset.

Coefficient	Attn				Joint			
	1	2	3	4	1	2	3	4
10^{-1}	28.56	29.85	31.37	33.47	28.15	29.71	31.35	33.11
10^{-2}	28.58	29.87	31.39	33.49	28.16	29.73	31.38	33.12
10^{-3}	**28.61**	**29.88**	**31.41**	**33.52**	28.16	29.73	**31.39**	**33.13**
10^{-4}	28.59	29.86	31.40	33.50	**28.17**	**29.74**	31.38	33.12
10^{-5}	28.58	29.84	31.38	33.49	28.16	29.74	31.37	33.11
Coefficient	Scale				Factorized			
	1	2	3	4	1	2	3	4
10^{-1}	27.63	29.35	31.09	32.93	26.90	28.19	29.60	31.26
10^{-2}	27.68	29.40	31.13	**32.96**	**26.91**	**28.21**	29.60	31.27
10^{-3}	27.70	29.42	**31.16**	32.93	26.91	28.20	**29.61**	31.28
10^{-4}	**27.73**	**29.44**	31.15	32.92	26.90	28.20	29.59	**31.28**
10^{-5}	27.70	29.43	31.14	32.92	26.90	28.19	29.58	31.27

Rectifier Learning Coefficient Exploration Result. Choosing the rectifier learning coefficients α in Eq. 9 is critical to the performance of the compression model. We empirically set an $\alpha_{max} = 10^{-1}$ and $\alpha_{min} = 10^{-5}$ for PSNR to cover feasible values of the learning coefficient. We explore learning coefficients between α_{max} and α_{min} following the exploration strategy in Sec. 3.3. The exploration for each coefficient value usually completes with 15 rounds of iterations. Tab. 2 shows the PSNR values for all models achieved at explored coefficients, where the coefficient producing the best PSNR (highlighted) is selected for predictive training. It is also observed that the PSNR performance monotonically degrades when the coefficient increases above or decreases below the selected value. Note that, as we round off the PSNR value to two decimal places, some adjacent rows with different coefficients may show the same PSNR value, e.g., 26.91 dB at $q = 1$ for coefficients 10^{-2} and 10^{-3} of Factorized. However, only the highlighted PSNR value is higher than others in the same column. The exploration of the rectifier learning coefficient efficiently identifies the optimal coefficient that ensures a better predictive training result than that without exploration.

Note that we do not repeat the same exploration process for MS-SSIM as we do for PSNR. The reason is we empirically find the distribution of optimal coefficients is similar for PSNR and MS-SSIM on different models, except that the optimal



(a) Model optimized for PSNR

(b) Model optimized for MS-SSIM

Figure 5. Quantization error reduction.

coefficient for MS-SSIM is roughly $10\times$ smaller than that for PSNR. As a result, we multiply the optimal coefficient in Tab. 2 by 0.1 for MS-SSIM during predictive training phase.

C. Training Configuration and Detailed Experimental Metrics

Training configuration. For enhanced models, the rectifier learning coefficient exploration and the STP training phases are conducted on the Flickr image dataset (Flickr, 2021) and the ImageNet dataset (Deng et al., 2009), respectively. Similar to the baseline training, the exploration and STP training phases adopt a batch size of 32. We fix the learning rate at 10^{-6} in both phases, which yields the best performance. The training is performed on a desktop with 2 NVIDIA RTX 3090 GPUs.

Metrics. As mentione in 4.1, we compare the performance in terms of rate and distortion trade-offs, measured by bpp and PSNR or MS-SSIM. To show the performance of our approach at different compression qualities, we repeat our experiments at compression quality levels $q \in \{1, 2, 3, 4\}$ of the codec models, where a greater value of q corresponds to a greater value of λ in Eq. 5. To demonstrate the capability of our approach in preserving the expressiveness of image features, we introduce a novel metric, the quantization error ϵ_Q. The quantization error is defined as the L2 distance between the input to the quantization operation (y) and the input to the decoder, which is either the result of quantization ($\hat{y}$) or QR ($\tilde{y}$) depending on whether a model is enhanced by QR, as shown in Eq. 12.

$$\epsilon_Q = \begin{cases} \|\tilde{y} - y\|_2 & \text{if using QR} \\ \|\hat{y} - y\|_2 & \text{if not using QR.} \end{cases} \tag{12}$$

D. Numerical Results of Coding Efficiency Improvement

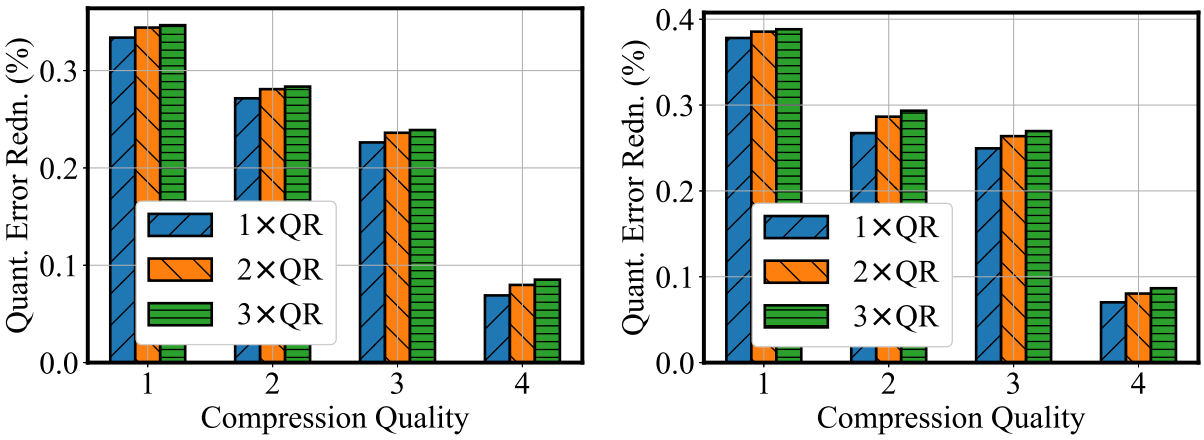
As mentioned in Sec. 4.2, Tab. 3 statistically shows the benefits of QR in terms of PSNR and MS-SSIM regarding the average and maximum values over all compression qualities.

Table 3. Image quality improvement in PSNR and MS-SSIM.

Metrics (dB)	Attn		Joint		Scale		Factorized	
	Avg	Max	Avg	Max	Avg	Max	Avg	Max
↑ PSNR	0.17±0.04	0.21	0.08±0.05	0.16	0.09±0.03	0.13	0.02±0.01	0.05
↑ MS-SSIM	0.19±0.05	0.25	0.12±0.01	0.12	0.11±0.05	0.19	0.12±0.03	0.15

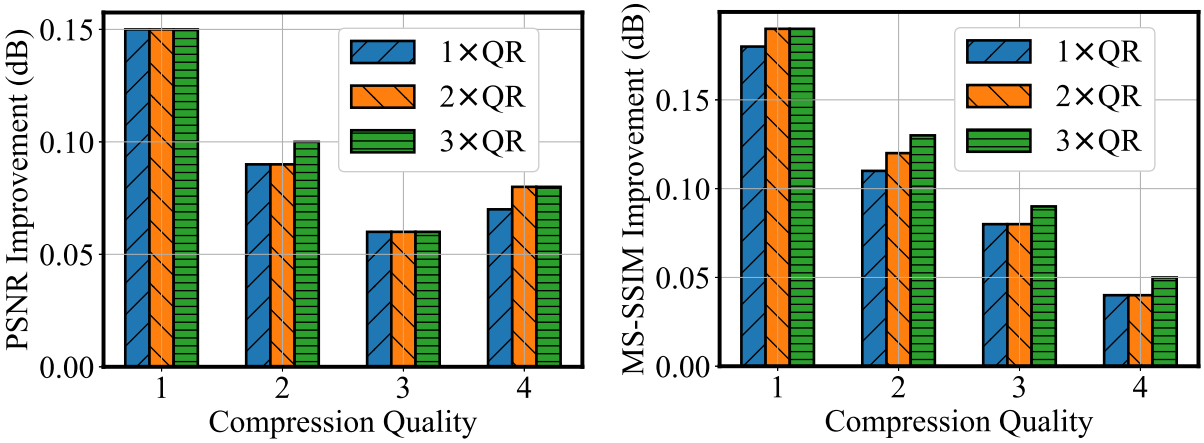
E. Evaluation of Quantization Error Reduction of QR

Quantization Error Reduction Result. Fig. 5 shows the reduction of quantization error by QR in percentage compared against all baseline models at various compression qualities. The reduction is generally more significant for more complex



(a) Model optimized for PSNR (b) Model optimized for MS-SSIM

Figure 6. Impact of different numbers of quantization rectifiers on quantization error reduction.



(a) Model optimized for PSNR (b) Model optimized for MS-SSIM

Figure 7. Impact of different numbers of quantization rectifiers on image quality.

models like Attn and Joint. Meanwhile, a lower compression quality tends to magnify the reduction. Notably, we observe the maximum quantization error reduction of 38% and 36% for Attn at compression quality $q = 1$, when the model is optimized for PSNR and MS-SSIM, respectively. We also notice models optimized for MS-SSIM, e.g., Factorized, exhibit randomness that causes the reduction to be slightly improved with a higher compression quality.

Analysis of Quantization Rectifier. With the performance gain of one QR, a natural idea is to embed multiple QRs into a codec. To this end, we apply multiple sequentially connected QRs to a baseline codec and analyze the improvements in image quality and quantization error reduction. Specifically, we compare the performance of one, two, and three QR(s), denoted by $1\times$QR, $2\times$QR, and $3\times$QR, respectively. Attn is used as the baseline model for different compression qualities. As Fig. 6 shows, the quantization error reduction is already significant when we have one QR, which is up to 35% for PSNR and 39% for MS-SSIM models. Despite more QRs further reduce the quantization error, the benefit of an additional QR, being less than 2%, is rather incremental in both PSNR and MS-SSIM. Similarly in Fig. 7, the improvement in the image quality measured by PSNR and MS-SSIM is significant with the first QR, which is up to 0.15 dB and 0.19 dB, respectively. After that, when we use two or three QRs, the image quality gain from one additional QR is at most 0.01 dB. When the compression quality changes, the above observations still apply. Given the fact that the computation and memory (storage) overhead of QR linearly increases with its number, affecting the training and encoding/decoding of the codec, we do not pursue a network design with more than one QR.