SANDWICHED IMAGE COMPRESSION: WRAPPING NEURAL NETWORKS AROUND A STANDARD CODEC

Onur G. Guleryuz, Philip A. Chou, Hugues Hoppe, Danhang Tang,
Ruofei Du, Philip Davidson, Sean Fanello

Google Research

{oguleryuz, philchou, hopp, danhangtang, ruofei, pdavidson, seanfa}@google.com

ABSTRACT

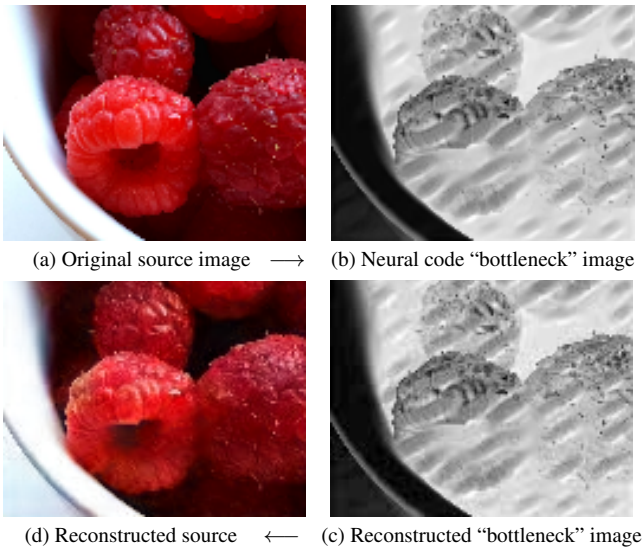
We sandwich a standard image codec between two neural networks: a preprocessor that outputs neural codes, and a postprocessor that reconstructs the image. The neural codes are compressed as ordinary images by the standard codec. Using differentiable proxies for both rate and distortion, we develop a rate-distortion optimization framework that trains the networks to generate neural codes that are efficiently compressible as images. This architecture not only improves rate-distortion performance for ordinary RGB images, but also enables efficient compression of alternative image types (such as normal maps of computer graphics) using standard image codecs. Results demonstrate the effectiveness and flexibility of neural processing in mapping a variety of input data modalities to the rigid structure of standard codecs. A surprising result is that the rate-distortion-optimized neural processing seamlessly learns to transport color images using a single-channel (grayscale) codec.

Index Terms— deep learning, image compression

1. INTRODUCTION

We explore a "sandwich" architecture for image compression, whereby a standardized image codec is placed between a neural preprocessor and a neural postprocessor. The neural preprocessor maps the original source image to latent variables forming neural images. We refer to these as "bottleneck" images because they are at the locus of the communication constraint. The bottleneck images are then compressed using standard codecs (*e.g.*, JPEG, HEVC, VP9 [1, 2, 3]), transmitted, and reconstructed by the standard decoder. The neural postprocessor converts the reconstructed bottleneck images to a reconstructed source image suitable for presentation.

We are primarily interested in scenarios where the neural processors accomplish processing well beyond conventional tasks like denoising and deblocking (Fig. 1). We show that the proposed sandwich architecture has several advantages: it improves the rate-distortion performance of the standard codec; it is able to adapt standard codecs to novel image types (*e.g.*, normal maps, multi-spectral images, medical images); and it leverages prior hardware and software investments in standard codecs.



(a) Original source image ⟶ (b) Neural code "bottleneck" image

(d) Reconstructed source ⟵ (c) Reconstructed "bottleneck" image

Fig. 1: Even with a simple codec (here, grayscale JPEG 4:0:0), a neural sandwich architecture can accomplish unconventional results. Whereas 4:0:0 usually just stores luminance, the neural preprocessor is able to encode a full RGB image into a single-channel bottleneck. The neural code image has low-frequency dither-like patterns that modulate color information yet also survive JPEG compression. At the decoding end, the neural postprocessor demodulates the patterns to recover the original color while also achieving deblocking. Fig. 6 shows another example and Fig. 7 presents rate-distortion results.

Some prior techniques augment standard codecs with neural networks. Neural preprocessing addresses denoising [4, 5, 6], and neural postprocessing is commonly used to reduce blocking and other coding artifacts [7, 8, 9]. In particular, Kim *et al.* [10] improve the output of the VVC/H266 intra-frame codec using residual dense networks (RDNs) and generative adversarial networks (GANs) to win the CVPR 2020 Learned Image Compression Challenge. In contrast our contribution is to consider a neural sandwich where the standard codec operates on images in a new space, *i.e.*, a latent space optimized for a particular class of input data.

End-to-end neural image compression [11]–[18] removes

the standard codec from the sandwich altogether and instead uses a uniform scalar quantizer for the latent variables in the bottleneck along with a learned entropy coder. This direction has culminated in the use of GANs in combination with perceptual distortion measures to reduce the bitrate by a factor of two or more relative to state-of-art standard codecs with no perceptual degradation [19]. However, the resulting end-to-end neural codec has high computational complexity, with over a million floating-point operations per pixel.

Most neural-network compression results focus on perceptual metrics, trying to generate pictures that look good to a casual observer. One of our primary applications is the transport of data where the human subjective assessment of the immediately decoded data is not paramount as the data undergoes further processing (*e.g.*, images storing surface-normal vectors used to relight shapes in computer graphics rendering) or undergoes scientific analysis (*e.g.*, hyper-spectral data.) Hence we focus in this paper on rate-distortion results where distortion is measured with the ℓ_2-norm, though our results are easily generalizable.

We leverage standard image codecs to perform the heavy lifting of compression. Such codecs are highly optimized and often implemented in hardware. The combination of neural preprocessing and postprocessing around the standard image codec offers the ability for the standard image codec to carry latent neural codes, rather than typical images. Not only does this configuration offer improved rate-distortion performance for natural RGB images, but it permits great flexibility in tailoring the standard codecs to carry alternative image types, such as normal maps for graphics, multi-spectral images for remote sensing, and medical images, as well as tailoring to alternative distortion measures. Some of these aspects are investigated in the following sections.

2. SANDWICH ARCHITECTURE

The sandwich architecture is shown in Fig. 2(a). An *original source image* S with one or more full-resolution channels is mapped by a neural preprocessor into one or more channels of latent codes. Each channel of latent codes may be full resolution or reduced resolution. The channels of latent codes are grouped into one or more *bottleneck images* B suitable for consumption by a standard image codec. The bottleneck images are compressed by the standard image encoder into a bitstream which is decompressed by the corresponding decoder into *reconstructed bottleneck images* $\hat{B}$. The channels of the reconstructed bottleneck images are then mapped by a neural postprocessor into a *reconstructed source image* $\hat{S}$.

The standard image codec in the sandwich is configured to *avoid* any color conversion or further subsampling. Thus, it compresses three full-resolution channels as an image in 4:4:4 format, one full-resolution channel and two half-resolution channels as an image in 4:2:0 format, and one full-resolution channel as an image in 4:0:0 (*i.e.*, grayscale) format — all without color conversion. Other combinations of channels are

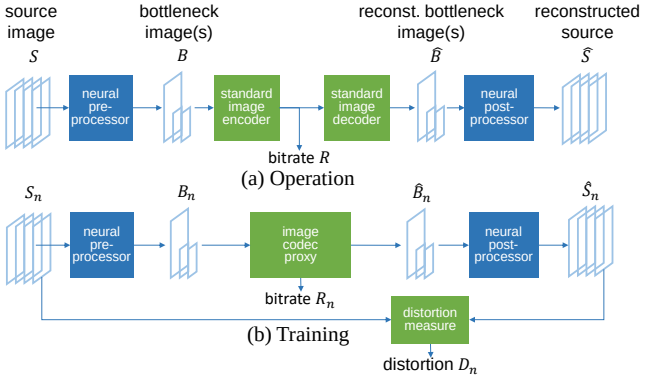


Fig. 2: Sandwich architecture in (a) operation and (b) training.

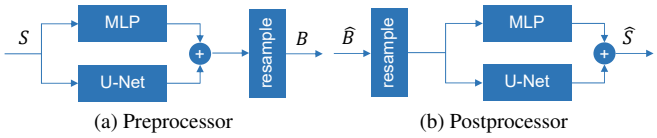


Fig. 3: Neural preprocessor and postprocessor.

processed by appropriate grouping.

Fig. 3 shows the network architectures for our neural preprocessor and postprocessor. The upper branch of the network learns pointwise operations, like color conversion, using a multilayer perceptron (MLP) or equivalently a series of 1×1 2D convolutional layers, while the lower branch uses a U-Net [20] to take into account more complex spatial context. At the output of the preprocessor, any half-resolution channels are obtained by sub-sampling, while at the input of the postprocessor, any half-resolution channels are first upsampled to full resolution using bilinear interpolation.

Fig. 2(b) shows the setup for training the neural preprocessor and postprocessor using gradient descent. Because derivatives cannot be back-propagated through the standard image codec, it is replaced by a differentiable *image codec proxy*. For each training example $n = 1, \ldots, N$, the image codec proxy reads the bottleneck image B_n and outputs the reconstructed bottleneck image $\hat{B}_n$, as a standard image codec would. It also outputs a real-valued estimate of the number of bits R_n that the standard image codec would use to encode B_n. The distortion is measured as the squared ℓ_2 error $D_n = ||S_n - \hat{S}_n||^2$ between the original and reconstructed source images. Together, the rate R_n and distortion D_n are the key elements of the differentiable loss function. Specifically, the neural preprocessor and postprocessor are optimized to minimize the Lagrangian $D + \lambda R$ of the average distortion $D = (1/N)\sum_n D_n$ and the average rate $R = (1/N)\sum_n R_n$.

The image codec proxy itself comprises the differentiable elements shown in Fig. 4. For convenience the image codec proxy is modeled after JPEG, an early codec for natural images. Nevertheless, experimental results show that it induces the trained preprocessor and postprocessor to produce bottleneck images sufficiently like natural images that they can also

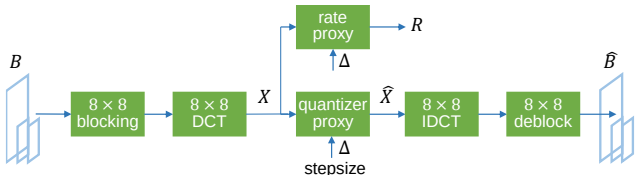
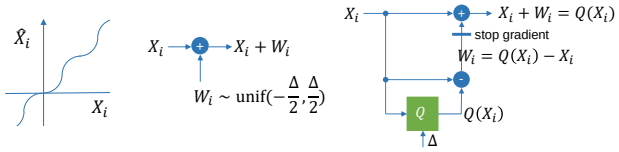


Fig. 4: Image codec proxy.



(a) Soft quantizer (b) Additive noise (c) Straight-through

Fig. 5: Possible quantizer proxies.

be compressed efficiently by other codecs such as HEVC. The image codec proxy spatially partitions the input channels, whether they are full-resolution or half-resolution, into 8×8 blocks. In the DCT domain, the blocks $X = [X_i]$ are processed independently, using (1) a "differentiable quantizer" (called a *quantizer proxy*) to create distorted DCT coefficients $\hat{X}_i = Q(X_i)$, and (2) a differentiable entropy measure (called a *rate proxy*) to estimate the bitrate required to represent the distorted coefficients $\hat{X}_i$. Both proxies take the nominal quantization stepsize Δ as an additional input.

Various differentiable **quantizer proxies** are possible (Fig. 5). Luo *et al.* [21] use a soft quantizer $Q(X_i)$ whose transfer characteristic is a third-order polynomial spline. Most end-to-end image compression works (*e.g.*, [11]–[17]) use either additive noise $Q(X_i) = X_i + W_i$ (where W_i is i.i.d., $W_i \sim \text{unif}(-\Delta/2, \Delta/2)$) or a "straight-through" quantizer which is a function $Q(X_i) = X_i + \text{stop\_gradient}(W_i)$, where $W_i = \Delta\text{round}(X_i/\Delta) - X_i$ is the true quantization noise and the stop_gradient$(\cdot)$ notation indicates gradients should not be propagated [22]. In all cases, the derivative of $\hat{X}_i = Q(X_i)$ with respect to X_i is nonzero almost everywhere. This lets us obtain non-trivial gradients of the end-to-end distortion $||S - \hat{S}||^2$ with respect to the parameters of the preprocessor using the chain rule and back-propagation. We use straight-through quantization in our experiments.

Various differentiable **rate proxies** are also possible. A convenient family of rate proxies $R(X)$ estimates the bitrate for a block of transform coefficients $X = [X_i]$ using affine functions of $\|X\|_2^2$, $\|X\|_1$, or $\|X\|_0$. We focus on the latter in our experiments, since it is shown in [23] that an affine function of the number of nonzero quantized transform coefficients, $R(X) = a\sum_i \mathbb{1}\{|x_i| \geq \Delta/2\} + b$, is an accurate rate proxy for transform codes. In our work, we approximate the indicator function $\mathbb{1}\{|x_i| \geq \Delta/2\}$ by the smooth differentiable function $\log(1 + |x_i|/\Delta)$. (An alternative would be to use $\tanh(|x_i|/\Delta)$.) In sum, our rate proxy for a bottleneck image $B = [X^{(k)}]$ comprising multiple blocks $X^{(k)}$ is

$$R(B) = \sum_k R(X^{(k)}) = a \sum_{k,i} \log\left(1 + \left|x_i^{(k)}\right| / \Delta\right) + b. \quad (1)$$

We set $b = 0$ and determine a for each bottleneck image B so that the rate proxy model matches the actual bitrate of the standard JPEG codec on that image, *i.e.*:

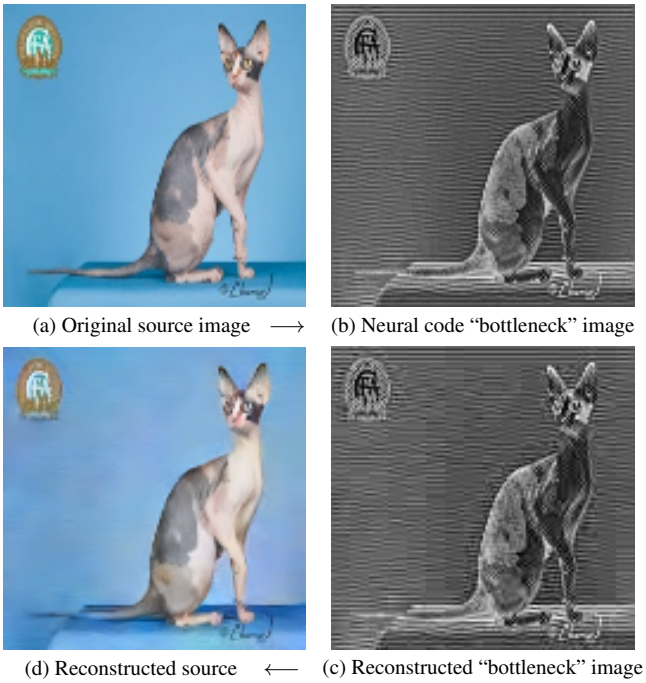
$$a = \frac{\text{JPEG}(B, \Delta)}{\sum_{k,i} \log\left(1 + \left|x_i^{(k)}\right| / \Delta\right)}. \quad (2)$$

This ensures that the differentiable function $R(B)$ is exactly equal to JPEG(B, Δ) and that proper weighting is given to its derivatives on a per-image basis. Any image codec besides JPEG can also be used. Similarly to the gradient of the distortion, the gradient of $R(B)$ with respect to the parameters of the preprocessor can be computed using back-propagation.

3. EXPERIMENTAL RESULTS

Our results include two RGB datasets [24, 25] and a computer graphics dataset [26]. Training and evaluation are performed on distinct subsets of each dataset. All reported results use the evaluation subsets with actual compression. U-Nets have a multi-resolution ladder of four with channels doubling up the ladder from 32 to 512. Convolutions are 3×3. MLP networks have two layers, both with three channels. We obtain R-D curves as follows. We train four models m_i with different Lagrange multiplier values λ_i using established $D + \lambda R$ optimization [27], letting the step-size Δ_i be a trained parameter. For each model, we obtain an R-D curve by encoding the images using a sweep over many step-size values. Finally we compute the Pareto frontier of these four curves.

Fig. 7 shows rate-distortion (R-D) results over the RGB evaluation set. For the 4:0:0 format, the standard codec can



(a) Original source image ⟶ (b) Neural code "bottleneck" image

(d) Reconstructed source ⟵ (c) Reconstructed "bottleneck" image

Fig. 6: Additional result as in Fig. 1.

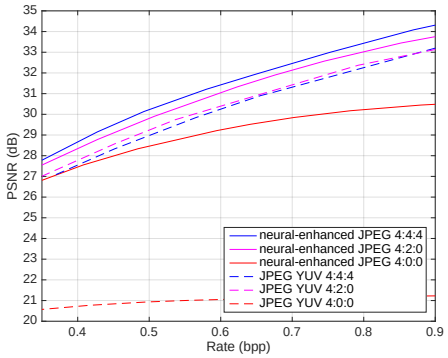


Fig. 7: JPEG rate-distortion results across 500 RGB test images, comparing the standard codec and neural-enhanced versions.

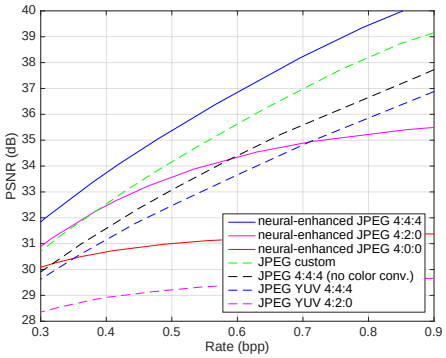


Fig. 8: JPEG rate-distortion on 500 normal-map images, comparing the standard codec and neural-enhanced versions.

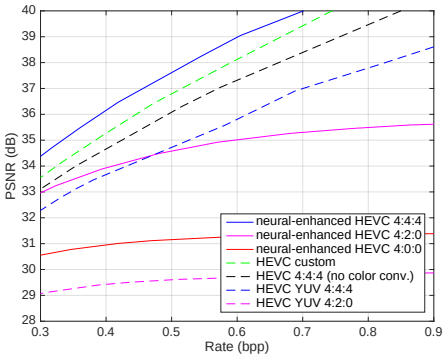
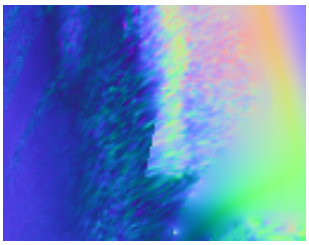
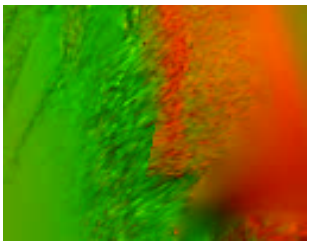


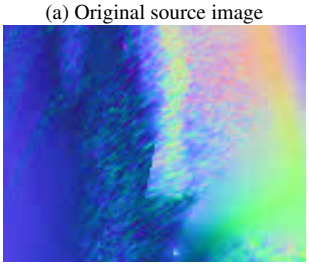
Fig. 9: HEVC rate-distortion on 500 normal-map images, comparing the standard codec and neural-enhanced versions.



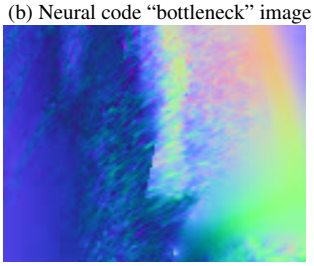
(a) Original source image



(b) Neural code "bottleneck" image



(c) JPEG 4:4:4 (RGB)



(d) Reconstructed source

Fig. 10: Compression of a normal-map image at 0.8 bits/pixel. JPEG 4:4:4 (RGB) achieves 32.2 dB, whereas the neural-enhanced 4:4:4 format attains 34.2 dB. Refer to Fig. 8 for full R-D results.

only transport grayscale, whereas the neural-enhanced version manages to transport color through modulating patterns exemplified in Fig. 1 and Fig. 6, resulting in substantial R-D improvements (6–9 dB). Results for the 4:2:0 and 4:4:4 formats are close to one another for the standard codec. The neural-enhanced codec not only performs better but finds a way to utilize the denser 4:4:4 sampling for improved gains. Observe also that at very low rates the neural enhanced 4:0:0 codec is competitive with JPEG YUV codecs.

Fig. 8 presents the R-D results for normal-map images, with one example image shown in Fig. 10. Each pixel stores a unit-norm vector $n = (n_x, n_y, n_z)$ representing the tangent-space normal ($n_z \geq 0$) with respect to a triangle mesh. We map each channel $[-1, 1]$ to RGB $[0, 255]$; note that the third channel n_z is redundant. The best results are obtained via the neural-enhanced full resolution codec which exploits the dependency. Neural-enhancing codecs with 4:2:0 and 4:0:0 formats provide significant gains over their respective baselines. JPEG YUV 4:2:0 and YUV 4:4:4 use RGB-YUV conversion, which does not provide any advantage for this dataset. JPEG 4:4:4 (no color conv.), which has no color transform, is better. The best non-neural result (JPEG custom) is obtained by zeroing out the third component n_z during compression, and recovering it in a postprocess as $\hat{n}_z = 1 - (\hat{n}_x^2 + \hat{n}_y^2)^{\frac{1}{2}}$. However, the best result overall (by more than 1 dB) uses a neural-enhanced codec with a 4:4:4 format.

Fig. 9 uses the same neural processing of Fig. 8, *i.e.*, the neural processing optimized for JPEG-like proxies, to sandwich HEVC without any retraining. Despite HEVC being a significantly different codec, similar results are obtained.[1]

4. CONCLUSION

The sandwiched, neural-enhanced codecs provide the greatest benefit when the input data space differs from that expected by the codec. In this paper we highlight scenarios where neural processing can have significant impact. A clear example is a tangent-space normal map, in which the vector stored at each pixel is constrained to lie on a hemisphere. However, sandwiched codecs should also benefit the compression of other image types, such as multispectral images in remote sensing, material maps in graphics, medical images, LIDAR images, depth images, HDR images, motion fields, and so forth. Thus, we envision that the sandwich architecture provides a general solution for adapting standard image and video codecs to the compression of new image types, and will further improve rate-distortion performance of these codecs on specific RGB images. The techniques mentioned could of course also be used in future standards as new coding modes for macroblocks or larger coding units. Directions for future work include applying the sandwich architecture to other image types, minimizing the complexity of the neural processing, and extending the work to video.

[1] JPEG/HEVC 4:0:0 results are omitted as they achieve only ∼11 dB.

5. REFERENCES

[1] William B. Pennebaker and Joan L. Mitchell, *JPEG Still Image Data Compression Standard*, Van Nostrand Reinhold, 1992.

[2] Mathias Wien, *High Efficiency Video Coding: Coding Tools and Specification*, Springer Publishing Company, Incorporated, 2014.

[3] Debargha Mukherjee, Jim Bankoski, Adrian Grange, Jingning Han, John Koleszar, Paul Wilkins, Yaowu Xu, and Ronald Bultje, "The Latest Open-Source Video Codec VP9 - An Overview and Preliminary Results," in *2013 Picture Coding Symposium (PCS)*. 2013, pp. 390–393, IEEE.

[4] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142–3155, 2017.

[5] Chunwei Tian, Lunke Fei, Wenxian Zheng, Yong Xu, Wangmeng Zuo, and Chia-Wen Lin, "Deep Learning on Image Denoising: an Overview," *Neural Networks*, vol. 131, pp. 251 – 275, 2020.

[6] Huy Vu, Gene Cheung, and Yonina C. Eldar, "Unrolling of Deep Graph Total Variation for Image Denoising," in *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 2050–2054.

[7] P. Svoboda, Michal Hradis, David Barina, and P. Zemcík, "Compression Artifacts Removal Using Convolutional Neural Networks," *ArXiv*, vol. abs/1605.00366, 2016.

[8] T. Kim, H. Lee, H. Son, and S. Lee, "SF-CNN: a Fast Compression Artifacts Removal Via Spatial-to-Frequency Convolutional Neural Networks," in *2019 IEEE International Conference on Image Processing (ICIP)*, 2019, pp. 3606–3610.

[9] Jun Niu, "End-to-End JPEG Decoding and Artifacts Suppression Using Heterogeneous Residual Convolutional Neural Network," *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2020.

[10] Y. Kim, S. Cho, J. Lee, S.-Y. Jeong, J. S. Choi, and J. Do, "Towards the Perceptual Quality Enhancement of Low Bit-Rate Compressed Images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2020, pp. 565–569.

[11] G. Toderici, S. M. O'Malley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, and R. Sukthankar, "Variable Rate Image Compression With Recurrent Neural Networks," in *4th Int. Conf. on Learning Representations (ICLR)*, 2016.

[12] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-End Optimization of Nonlinear Transform Codes for Perceptual Quality," in *2016 Picture Coding Symposium (PCS)*, 2016, pp. 1–5.

[13] G. Toderici, D. Vincent, N. Johnston, S. J. Hwang, D. Minnen, J. Shor, and M. Covell, "Full Resolution Image Compression With Recurrent Neural Networks," in *2017 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[14] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-End Optimized Image Compression," in *5th Int. Conf. on Learning Representations (ICLR)*, 2017.

[15] Johannes Ballé, "Efficient Nonlinear Transforms for Lossy Image Compression," in *2018 Picture Coding Symposium (PCS)*, 2018, pp. 248–252.

[16] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational Image Compression With a Scale Hyperprior," in *6th Int. Conf. on Learning Representations (ICLR)*, 2018.

[17] D. Minnen, J. Ballé, and G. Toderici, "Joint Autoregressive and Hierarchical Priors for Learned Image Compression," in *Advances in Neural Information Processing Systems 31*, 2018.

[18] J. Balle, P. A. Chou, D. Minnen, S. Singh, N. Johnston, E. Agustsson, S. J. Hwang, and G. Toderici, "Nonlinear Transform Coding," *IEEE Journal of Selected Topics in Signal Processing*, pp. 1–1, 2020.

[19] Fabian Mentzer, George D Toderici, Michael Tschannen, and Eirikur Agustsson, "High-Fidelity Generative Image Compression," in *Advances in Neural Information Processing Systems*, 2020, vol. 33, pp. 11913–11924.

[20] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015*, Cham, 2015, pp. 234–241.

[21] X. Luo, H. Talebi, F. Yang, M. Elad, and P. Milanfar, "The Rate-Distortion-Accuracy Tradeoff: JPEG Case Study," 2020.

[22] "Tensorflow API: tf.stopgradient," https://www.tensorflow.org/api_docs/python/tf/stop_gradient.

[23] Z. He and S. Mitra, "A Unified Rate-Distortion Analysis Framework for Transform Coding," *IEEE Trans. Circuits Syst. Video Technol*, vol. 11, pp. 1221–1236, 2001.

[24] "Dataset for the Challenge on Learned Image Compression 2020," http://www.tensorflow.org/datasets/catalog/clic.

[25] "The Oxford-IIIT Pet Dataset," https://www.tensorflow.org/datasets/catalog/oxford_iiit_pet.

[26] K. Guo, P. Lincoln, P. Davidson, J. Busch, X. Yu, M. Whalen, G. Harvey, S. Orts-Escolano, R. Pandey, J. Dourgarian, D. Tang, A. Tkach, A. Kowdle, E. Cooper, M. Dou, S. Fanello, G. Fyffe, C. Rhemann, J. Taylor, P. Debevec, and S. Izadi, "The Relightables: Volumetric Performance Capture of Humans With Realistic Relighting," in *ACM TOG*, 2019.

[27] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, 1992.