

Neural JPEG: End-to-End Image Compression Leveraging a Standard JPEG Encoder-Decoder

Ankur Mali[★], Alexander G. Ororbia[†], Daniel Kifer[★], C. Lee Giles[★]

[★] The Pennsylvania State University, University Park, PA, 16802, USA

[†]Rochester Institute of Technology, Rochester, NY, 14623, USA

Abstract

Recent advances in deep learning have led to superhuman performance across a variety of applications. Recently, these methods have been successfully employed to improve the rate-distortion performance in the task of image compression. However, current methods either use additional post-processing blocks on the decoder end to improve compression or propose an end-to-end compression scheme based on heuristics. For the majority of these, the trained deep neural networks (DNNs) are not compatible with standard encoders and would be difficult to deploy on personal computers and cellphones. In light of this, we propose a system that learns to improve the encoding performance by enhancing its internal neural representations on both the encoder and decoder ends, an approach we call Neural JPEG. We propose frequency domain pre-editing and post-editing methods to optimize the distribution of the DCT coefficients at both encoder and decoder ends in order to improve the standard compression (JPEG) method. Moreover, we design and integrate a scheme for jointly learning quantization tables within this hybrid neural compression framework. Experiments demonstrate that our approach successfully improves the rate-distortion performance over JPEG across various quality metrics, such as PSNR and MS-SSIM, and generate visually appealing images with better color retention quality.

Introduction

Over the years there has been a rapidly increasing use of the world wide web, propelled by the ever growing popularity of social media, to transmit visual signals, e.g., pictures and video, of ever rising resolution and quality. As a result, there is an equally increasing need to improve our ability to compress visual data.

Recently, in response to this problem, research in deep neural networks (DNNs) has begun to turn its attention to improving the rate-distortion performance of image compression frameworks. Current efforts have presented impressive results with end-to-end image compression systems using DNNs in a variety of ways [1, 2, 3]. Although powerful, these systems require carefully designing and training effective encoding/decoding functions as well as quantizers. Other methods perform post-editing and apply DNNs in an attempt to reduce the compression artifacts on the decoder end [4, 5]. Though this set of approaches has yielded impressive results, they unfortunately require a specifically trained decoder during the post-processing stage or a complex DNN-based decoder. As such they are not supported by the commonly used image viewers in most computers and smartphones. In addition, there is no guarantee that the compression results uncovered would hold when the input data distribution shifts, e.g., images of a completely different kind are presented to the system, presenting a challenge to DNN-driven approaches. Recent approaches that craft

hybrid decoders [6, 7, 8] have presented a very promising direction, yet they struggle to operate well at the lowest bit rate, given that the quantized signals they must work with to reconstruct the original input signal are extremely sparse. Another promising alternative is to design a hybrid encoder that enhances encoder signals resulting in better compression even at the lowest bit rates [9, 10]. However, these methods fail to remove artifacts at the decoder end, thus compromising compression quality in various situations.

In the spirit of the recent efforts to construct DNN-driven compression system, this paper proposes a simple approach to overcome the above mentioned shortcomings by handcrafting an image encoder-decoder system that exploits the inherent sparsity in quantization space while plugging into an existing image method. Specifically, we focus on the most common and widely-used standard image compression method, JPEG, though our framework is general enough that it could be leveraged/extended to work with others, e.g., JP2 or PGF. In this study, our approach improves the rate-distortion performance of JPEG by enhancing the latent representations acquired by its neural encoder and decoder by efficiently learning the DCT coefficients while at the same time ensuring that the bitstreams are decodable even with a standard decoder. Specifically, we construct a system that leverages an encoder and decoder that are each driven by sparse recurrent neural networks (SMRNNs) trained within the effective framework of neural iterative refinement [6] – the recurrent encoder learns to “pre-edit” an input image in the frequency domain (producing values that serve as the necessary DCT coefficients) while the recurrent decoder learns to reduce artifacts in the reconstructed image. To further boost our rate-distortion performance at the lowest bitrate, we overcome the limitations of handcrafted codecs built into JPEG and JPEG-2000, which rely on quantization driven by fixed transformation matrices and entropy encoding, by developing a scheme that jointly learns the quantization table with the recurrent encoder. As a result, since our approach learns the quantization table, the entire hybrid system could be viewed as a differentiable JPEG pipeline.

In summary, our contributions are as follows:

- We extend on prior work and improve system rate-distortion performance by optimizing the JPEG encoder in the frequency domain.
- We facilitate better coefficient construction at the decoder end by optimizing the JPEG decoder.
- A sparse recurrent network (Neural JPEG) is adapted to learn how to edit the DCT coefficients at both decoder and encoder ends.
- A learnable quantization table that is optimized jointly with the sparse recurrent encoder/decoder to improve rate-distortion performance, yielding an end-to-end, differentiable JPEG compression system.

Related Work

Widely-used lossy image compression methods such as JPEG and JPEG-2000 (JP2) employ a combination of fixed transformations using entropy-based encodings to achieve better compression [11]. This is suitable for real-time processing when memory and computational efficiency are needed. Recently, DNN approaches have been

shown to outperform these traditional methods in the task of image compression [12, 13]. However, most of this work has focused on designing end-to-end systems that reconstruct images in a two-dimensional space using architectural building-block models [14] such as auto-encoders, convolutional networks, and recurrent networks. Some early work crafted a framework based on variational autoencoders [15] that results in improved rate-distortion. Other complementary work [16, 17, 18], which outperformed classical techniques (at low bit rates) without harming perceptual quality, set the widely-adopted practice for using deep artificial neural networks (ANNs) in compression. Later methods that followed focused on using convolutional networks or generative adversarial networks (GANs) [19, 1, 20, 21]. Recent work has used spatial-temporal energy compaction [2], other energy compaction-based techniques [22], and filter-bank-based convolution networks [23]. Most of these end-to-end solutions have been designed to extract better latent representations and/or eliminate redundancies in the compression process. A more straightforward compression approach considered redundancy at the decoder side of the system and attempted to decompress by designing an iterative hybrid recurrent decoder [6, 7, 8]. Similarly, a standard encoder can be replaced with another DNN to enhance the model’s internal neural representations and decode information while only using a standard decoder both in the pixel [24] and frequency domains [10].

Neural JPEG

0.1 The JPEG Algorithm

We start by briefly introducing the workflow of the JPEG algorithm. The first step employs conversion of the input image from RGB to the YCbCr colorspace. Next, the image \mathbf{I} (of size $N \times M$ pixels) is divided into N non-overlapping blocks of size $N_i \times M_i$ pixels. The discrete cosine transform (DCT) is then applied to convert each block into the frequency domain. We denote the DCT coefficients of any given block (n, m) , where $n \in 0 \dots (N/N_i)$ and $m \in 0 \dots (M/M_i)$, for the luminance channel Y with $\mathbf{F}^{(Y)} = \mathbf{I}[n, m] \in \mathbb{R}^{8 \times 8}$ and accordingly for chrominance channels Cb, Cr . Furthermore, the DCT or DWT (in JPEG 2000) coefficients are quantized using two quantization tables: $\mathbf{Q}^{(L)} \in \mathbb{R}^{8 \times 8}$ for the luminance channel Y and $\mathbf{Q}^{(C)} \in \mathbb{R}^{8 \times 8}$ for the chrominance channels Cb, Cr followed by a rounding function:

$$\hat{\mathbf{Z}}_{u,v}^{(Y)} = \left\lfloor \frac{\mathbf{F}_{u,v}^{(Y)}}{Q_{u,v}^{(L)}} \right\rfloor, \quad \hat{\mathbf{Z}}_{u,v}^{(Cb)} = \left\lfloor \frac{\mathbf{F}_{u,v}^{(Cb)}}{Q_{u,v}^{(C)}} \right\rfloor, \quad \hat{\mathbf{Z}}_{u,v}^{(Cr)} = \left\lfloor \frac{\mathbf{F}_{u,v}^{(Cr)}}{Q_{u,v}^{(C)}} \right\rfloor, \quad \text{for } u, v \in [1, 8]. \quad (1)$$

Finally, these quantized DCT coefficients are passed to an entropy coding module to finish compressing the input image.

0.2 The Neural JPEG Architecture

Our complete Neural JPEG architecture, which integrates elements from [25] and [6], leverages a recurrent encoder model $E_{\Theta}(\mathbf{I})$ that drives/interacts with the encoder of standard JPEG, a differentiable quantization table and rounding module, and a recurrent decoder model $D_{\Theta}(\hat{\mathbf{Z}})$ that drives/interacts with the decoder of standard JPEG.

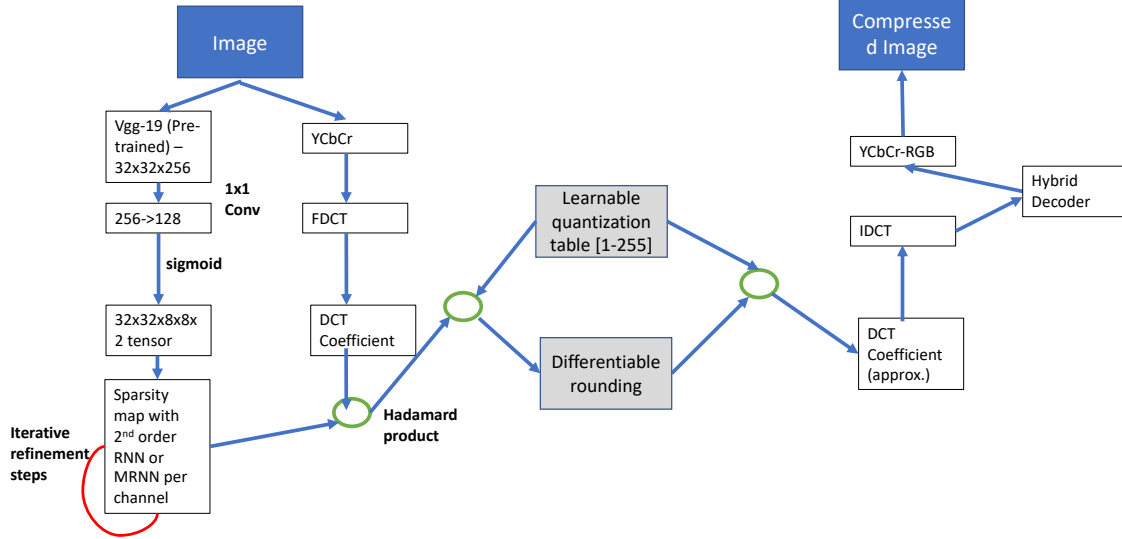


Figure 1: The end-to-end Neural JPEG architecture for image compression.

All of these components/modules are optimized jointly yielding a one-time training end-to-end differentiable JPEG pipeline. A graphical representation of the proposed Neural JPEG is depicted in Figure 1. Formally, the Neural JPEG architecture can be summarized as the following functions:

$$\hat{\mathbf{Z}} = E_{\theta}(\mathbf{I}), \quad \hat{\mathbf{I}} = D_{\theta}(\hat{\mathbf{Z}}) \implies \hat{\mathbf{I}} = D_{\theta}(E_{\theta}(\mathbf{I})). \quad (2)$$

where $\hat{\mathbf{I}}$ and $\hat{\mathbf{Z}}$ represents the reconstructed image and quantized DCT coefficients respectively.

Editing with Sparse RNNs: Inspired by [26] and [10], we design an approach that uses a neural models to either pre-edit (or iteratively process) an image \mathbf{I} before the quantization step of JPEG or post-edit the inverse DCT coefficients before converting back to the reconstructed image $\hat{\mathbf{I}}$. Specifically, for pre-editing, our neural encoder $E_{\theta}(\mathbf{I})$ proceeds according to the following steps to produce a set of “edit” weights:

1. Run \mathbf{I} through vision model $\mathbf{H}' = \text{DS}(f_V(\mathbf{I}; \Theta_V))$, where $\text{DS}()$ is the down-sampling operator. Note that f_V in this paper was chosen to be a pre-trained VGGNet-19 [27] and Θ_V are its parameters. A 1×1 convolution is then applied to \mathbf{H}' to reduce its channel dimension to 128 and the final output is $\mathbf{H} = \sigma(\mathbf{H}')$ where σ is the sigmoid activation function to limit the output values to the range $[0, 1]$. The 128 output channels of \mathbf{H} are finally reshaped to a $8 \times 8 \times 2$ tensor for each block which is then split into two 8×8 matrices (one for luminance and one for chrominance), i.e., \mathbf{H}_L and \mathbf{H}_C .
2. Use the sparse multiplicative RNN (SM-RNN) component to process both \mathbf{H}_L and \mathbf{H}_C (in parallel) for K steps (which is the process of iterative refinement, adapted from [6], using the multiplicative RNN model proposed in [28]). This

SM-RNN produces the desired activation map according to the dynamics:

$$\mathbf{f}_k^L = \text{diag}(\mathbf{W}_f \mathbf{H}_L) \cdot (\mathbf{V}_f \mathbf{z}_{k-1}^L) \quad (3)$$

$$\mathbf{z}_k^L = \phi(\mathbf{V}_z \mathbf{f}_k^L + \mathbf{W}_z \mathbf{H}_L), \text{ where } k = 0, 1, \dots, K \quad (4)$$

where the above equation is simply repeated for \mathbf{H}_C (just replace L in Equation 4 with C) and $K = 3$ in this paper. The final set of sparse “edit” values are produced via the following:

$$\mathbf{c}_L = \text{kWTA}(\mathbf{U} \mathbf{z}_K^L), \mathbf{c}_C = \text{kWTA}(\mathbf{U} \mathbf{z}_K^C) \quad (5)$$

where $\text{kWTA}()$ is the k winners-take-all function (setting all values that are less than the k greatest neurons are set to zero).

Note that the above is repeated for each of the $N \cdot M$ blocks. Before applying our quantization table, we multiply each DCT coefficient produced by the original JPEG encoder by its corresponding edit score obtained from the above process.

For the decoder end of our system, we utilize a SM-RNN similar to the encoder process described above but tie its weights to those of the SM-RNN encoder module (both models would have the same dimensionalities for their parameters given they operate in the same internal latent space). In short, after applying the inverse DCT transform to the outputs of the quantization table and rounding modules (which produce $\hat{\mathbf{Z}}$), the SM-RNN decoder takes in the $\hat{\mathbf{Z}}$, processes it K times (in a process similar to the one depicted above) and finally produces the reconstructed image $\hat{\mathbf{I}}$.

A Learnable Quantization Table and Rounding Module: In line with prior work [10] we also use the differentiable JPEG pipeline to learn the quantization tables. We replace attention mechanism with sparse RNN to better capture the importance of each representation associated with each channel. We use $Q_\theta^{(L)}$ and $Q_\theta^{(C)}$ as optimization variables for luminance and chrominance. The range of quantization table values are clipped to $[1, 255]$ which helps in optimization process. Our overall approach in this stage is as follows:

$$\begin{aligned} \hat{\mathbf{Z}}^{(Y)}[n, m] &= \left[\mathbf{F}^{(Y)}[n, m] \odot \text{SMRNN}^{(L)}[n, m] \odot \bar{\mathbf{Q}}_\theta^{(L)} \right]_{\text{approx}} \\ \hat{\mathbf{Z}}^{(Cr)}[n, m] &= \left[\mathbf{F}^{(Cr)}[n, m] \odot \text{SMRNN}^{(C)}[n, m] \odot \bar{\mathbf{Q}}_\theta^{(C)} \right]_{\text{approx}} \\ \hat{\mathbf{Z}}^{(Cb)}[n, m] &= \left[\mathbf{F}^{(Cb)}[n, m] \odot \text{SMRNN}^{(C)}[n, m] \odot \bar{\mathbf{Q}}_\theta^{(C)} \right]_{\text{approx}} \end{aligned} \quad (6)$$

for $n \in [1, N], m \in [1, M]$,

with $\bar{Q}_{u,v}^{(L)} = \frac{1}{Q_{u,v}^{(L)}}$, $\bar{Q}_{u,v}^{(C)} = \frac{1}{Q_{u,v}^{(C)}}$, for $u, v \in [1, 8]$.

Here \odot represents hadamard product. This The modification introduced above loses some features and is not recoverable at decoder end. Multiplying the DCT coefficients by a number ≤ 1 acts like a frequency filter, suppressing the higher frequency

to get low-pass filter. By combining sparse weights with the DCT-coefficient we get a smoothing filter that is spatially adaptive and also applicable across various frequencies.

For the rounding module, we remove the entropy encoding used in JPEG and replace the hard rounding operation with a differentiable 3rd order approximation:

$$\lfloor \hat{\mathbf{Z}} \rfloor_{approx} = \lfloor \hat{\mathbf{Z}} \rfloor + (\lfloor \hat{\mathbf{Z}} \rfloor - \hat{\mathbf{Z}})^3. \quad (7)$$

0.3 Loss Formulation

For designing end-to-end framework one needs to find a efficiently optimize for rate-distortion tradeoff. Additionally we introduced alignment loss that is responsible for controlling hybrid decoder. For any given input image x and the reconstructed image \hat{x} , and learned parameters θ , our loss function has the general form as follows:

$$\begin{aligned} \mathcal{L}(x, \hat{x}; \theta) &= \lambda \cdot d(x, \hat{x}) + (1 - \lambda - 0.01)r(x, \hat{x}; \theta) + 0.01al(x, \hat{x}), \\ &\text{with } x, \hat{x} \in \{t \in \mathbb{R} \mid 0 \leq t \leq 255\}^{8N \times 8M \times 3}, \quad \lambda \in \mathbb{R}^+, \end{aligned} \quad (8)$$

where $al(x, \hat{x}; \theta)$ is alignment loss, $r(x, \hat{x}; \theta)$ is rate loss and $d(x, \hat{x})$ is distortion loss. The parameter λ determines the ratio of the triplet loss and hence balances alignment, distortion and rate. The ideal value for λ is obtain based on validation performance.

0.3.1 The Distortion Loss

The distortion loss is responsible for measuring similarity between compressed and original images. To achieve this we use the combination MSE and LPIPS as follows:

$$d(x, \hat{x}) = \text{MSE}(x, \hat{x}) + \gamma \cdot \text{LPIPS}(x, \hat{x}) \quad (9)$$

where we introduce γ as the LPIPS modulating factor.

0.3.2 The Rate Loss

We use the rate loss formulation proposed by [10] by replace attention map with sparse map obtained from SMRNN that is represented as follows:

$$r(x; \theta) = \alpha(\|\bar{Q}_\theta^{(L)}\|_1 + \|\bar{Q}_\theta^{(C)}\|_1) + \beta(\text{mean}(\text{SMRNN}_\theta^{(L)}(x)) + \text{mean}(\text{SMRNN}_\theta^{(C)}(x))) \quad (10)$$

with the mean function: $\text{mean}(\text{SMRNN}) = \frac{1}{|\mathcal{P}|} \sum_{\vec{p} \in \mathcal{P}} \text{SMRNN}_{\vec{p}}$, where \mathcal{P} is the index set over all entries in the tensor SMRNN .

0.3.3 Alignment Loss

The alignment loss is ensuring signals are robust at decoder end. To achieve this we propose using combination of MSE and Mean Absolute Error (MAE) as follows:

$$al(x, \hat{x}) = (1 - \sigma)\text{MSE}(x, \hat{x}) + \sigma \cdot \text{MAE}(x, \hat{x}) \quad (11)$$

where $\sigma = [0.1 - 0.4]$ (values chosen based on validation set). Based on the triplet loss definitions above we can pose the optimization objective as: $\min_{\theta} \mathcal{L}(x, D_\theta(E_\theta(x)); \theta)$.

Experiments

0.4 Evaluation Metrics

We use widely used compression optimization metric for measuring image similarity the Mean Squared Error (MSE) and the Peak Signal to Noise Ratio (PSNR). Similarly to [29, 10] we define the MSE and PSNR for the tensors $x, \hat{x} \in hcalX$ of arbitrary dimension as follows (x is also the input image \mathbf{I}):

$$\begin{aligned} \text{MSE}(x, \hat{x}) &= \frac{1}{|\mathcal{P}|} \sum_{\vec{p} \in \mathcal{P}} (x_{\vec{p}} - \hat{x}_{\vec{p}})^2 \\ \text{PSNR}(x, \hat{x}) &= 10 \log_{10} \left(\frac{255^2}{\text{MSE}(x, \hat{x})} \right) \end{aligned} \tag{12}$$

where \mathcal{P} is the set of pixel indices and $x_{\vec{p}}, \hat{x}_{\vec{p}} \in [0, 255]$, $\forall \vec{p} \in \mathcal{P}$. To better measure the visual appeal of images we also use the Multi-Scale Structural Similarity (MS-SSIM) [30], converted to a logarithmic scale as follows:

$$\text{MS-SSIM [dB]} = -10 \log_{10}(1 - \text{MS-SSIM}) \tag{13}$$

Additionally, we also use DNN-friendly the Learned Perceptual Image Patch Similarity (LPIPS) [31] objective function.

0.5 Datasets and Training Procedure

The Neural JPEG network is trained on the dataset provided in prior work [32, 10]. It consists of 3640 HDR images. For training, we use the merged HDR images and extract image patches of size 256 obtained from random cropping. We follow the same extraction process and experimental protocol proposed by [10] We evaluate our model on the Kodak dataset, consisting of 24 uncompressed images of size 768×512 . Additionally, we validate our model on validation set from DIV2K [33, 34] containing 100 high quality images with 2040 pixels. Our model is optimized using Adam with initial learning rate 1.0 reduced to 10^{-8} using polynomial decay. We use batch size of 32 for all experiments and performed grid search to find optimal hidden sizes for SMRNN, sparsity level k , and λ . We use pre-trained VGG-19 model (trained on ImageNet) and fine-tune these layers while training. The 1×1 convolutional layer is initialized using orthogonal matrices. We follow prior experimental protocol [10], hence the quantization table variables in this work are also initialized uniformly in the interval $[1s, 2s]$ and are limited to be in the range $[1s, 255s]$. Where the scaling factor is always $s > 0$ and in this experiments is set to is a $s = 10^{-5}$. Then we can get the final quantization tables by multiplying factor by s^{-1} . We use standard evaluation metrics such as PSNR, MSE, MS-SSIM to report our model performance.

Result and Discussion

We evaluated our model on 2 out of 6 benchmarks used in prior work [6] using 3 metrics [35]. These metrics are Peak Signal to Noise Ratio (PSNR), structural similarity

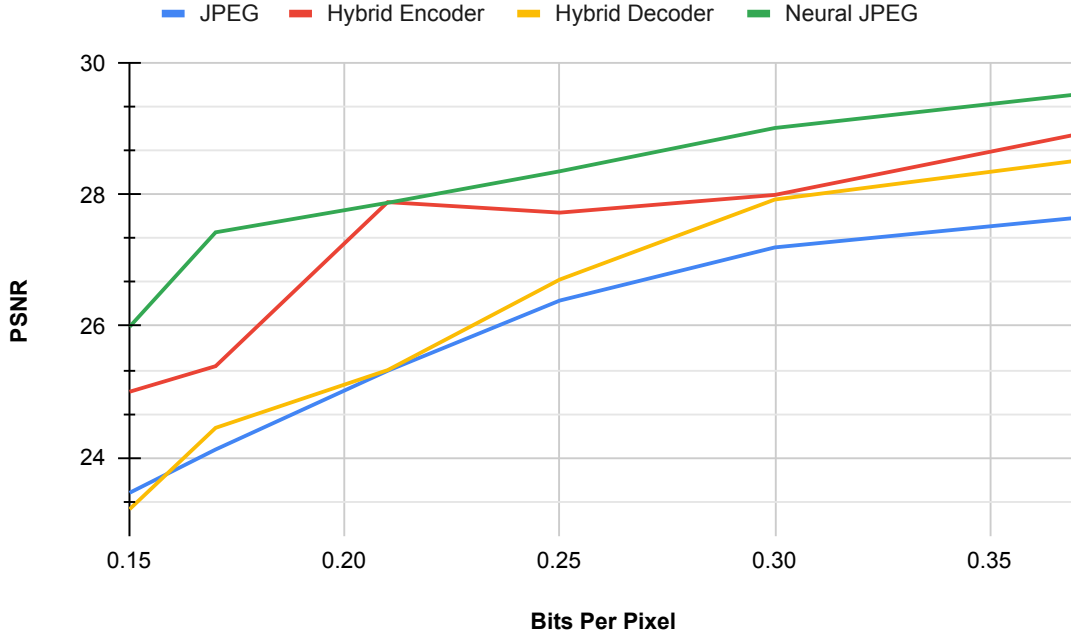


Figure 2: Performance of various compression model with various bit rate setting

(SSIM), and multi-scale structural similarity (MS-SSIM [36], or MS^3IM . We compare our model against wide variety of compression approaches such pure neural-based GOOG[17] and E2E [12]. We also compare to the models of [6] and [10]. Results are reported in Table 2 – we see all models perform stably, however, when the bit rates are reduced (see Table 1) all hybrid models start struggling, whereas neural-based models stay consistent. This suggests that the performance of these hybrid models is limited due to the fixed map used by JPEG. This support our hypothesis that at lower bit rates enhancing JPEG signals further improves performance and that Neural JPEG does this efficiently. Our model does struggles due to some artifacts, but these can be removed with JPEG artifact removal and we note that our overall model ensures the structure/content of image is kept intact. In other words, whenever we operate at the lowest bit rates (see Figure 2), JPEG completely loses image information while the Neural JPEG makes up the difference since it can recover majority of missing chunks of features. Since JPEG quality based on PSNR at 0.25bpp is equivalent to Neural JPEG image quality at 0.19bpp, we conclude that even at the lowest bit rates, our model tries to remember majority of the signal.

Conclusions and Future Work

Our experiments show that our approach, Neural JPEG, improves JPEG encoding and decoding through sparse RNN smoothing and learned quantization tables that are trained end-to-end in an differentiable framework. The proposed model leads to better compression/reconstruction at lowest bit rates when evluated using metrics

Table 1: Test results for Kodak (bpp 0.38), 8-bit Compression Benchmark (CB, bpp, 0.371)

Model	Kodak			CB 8-Bit		
	PSNR	SSIM	MS^3IM	PSNR	SSIM	MS^3IM
<i>JPEG</i>	31.2190	0.7412	0.9011	32.7893	0.7921	0.9009
<i>GOOG-JPEG</i>	30.9821	0.7415	0.9016	31.899	0.7967	0.9012
<i>E2E (Neural)</i>	30.3471	0.7521	0.9021	31.769	0.8001	0.9016
<i>MLP-JPEG</i>	27.8325	0.8399	0.9444	27.8089	0.8371	0.9475
<i>Δ-RNN-JPEG</i>	28.5093	0.8411	0.9487	28.0461	0.8403	0.9535
<i>GRU-JPEG</i>	28.5081	0.8400	0.9474	28.0446	0.8379	0.9533
<i>Hybrid Decoder - JPEG</i>	31.1282	0.7413	0.9011	32.7100	0.7920	0.9010
<i>Hybrid Encoder</i>	31.512	0.7520	0.9021	31.6712	0.8002	0.9016
<i>Neural JPEG (Ours)</i>	31.732	0.7521	0.9022	32.400	0.7926	0.9012

Table 2: Test results for Kodak (bpp 0.37), 8-bit Compression Benchmark (CB, bpp, 0.341)

Model	Kodak			CB 8-Bit		
	PSNR	SSIM	MS^3IM	PSNR	SSIM	MS^3IM
<i>JPEG</i>	27.6540	0.7733	0.9291	27.5481	0.8330	0.9383
<i>GOOG-JPEG</i>	27.9613	0.8017	0.9557	27.8458	0.8396	0.9562
<i>E2E (Neural)</i>	28.9420	0.8502	0.9600	28.0999	0.8396	0.9562
<i>MLP-JPEG</i>	27.8325	0.8399	0.9444	27.8089	0.8371	0.9475
<i>Δ-RNN-JPEG</i>	28.5093	0.8411	0.9487	28.0461	0.8403	0.9535
<i>GRU-JPEG</i>	28.5081	0.8400	0.9474	28.0446	0.8379	0.9533
<i>Hybrid Decoder - JPEG</i>	28.5247	0.8409	0.9486	28.0461	0.8371	0.9532
<i>Hybrid Encoder</i>	28.8920	0.8411	0.9488	27.9211	0.8374	0.9534
<i>Neural JPEG (Ours)</i>	29.5247	0.8413	0.9489	27.8009	0.8375	0.9535

such as MSE, PSNR and also using perceptual metrics (LPIPS, MS-SSIM) that are known to be much closer to human perception. Most importantly, the improved encoder-decoder remains entirely compatible with any standard JPEG algorithm but produces significantly better colors than standard JPEG. We have shown that we can achieve improvement without directly estimating the entropy of the DCT coefficients, only regularizing the sparse maps and quantization tables. In the future, we wish to design an improved decoder that learns quantized signals from each color channel and uses a distribution-specific quantization table instead of a single differentiable quantization table.

References

- [1] Oren Rippel and Lubomir D. Bourdev, “Real-time adaptive image compression,” in *ICML*, 2017, pp. 2922–2930.
- [2] Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto, “Learning image and video compression through spatial-temporal energy compaction,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 10071–10080.

- [3] Ren Yang, Fabian Mentzer, Luc Van Gool, and Radu Timofte, “Learning for video compression with hierarchical quality and recurrent enhancement,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [4] Tingting Wang, Mingjin Chen, and Hongyang Chao, “A novel deep learning-based method of improving coding efficiency from the decoder-end for HEVC,” in *Proceedings of the Data Compression Conference (DCC)*. IEEE, 2017, pp. 410–419.
- [5] Tingting Wang, Wenhui Xiao, Mingjin Chen, and Hongyang Chao, “The multi-scale deep decoder for the standard HEVC bitstreams,” in *Proceedings of the Data Compression Conference (DCC)*. IEEE, 2018, pp. 197–206.
- [6] A. G. Ororbia, A. Mali, J. Wu, S. O’Connell, W. Dreese, D. Miller, and C. L. Giles, “Learned neural iterative decoding for lossy image compression systems,” in *DCC*, March 2019, pp. 3–12.
- [7] A. Mali, A. G. Ororbia, and C. L. Giles, “The sibling neural estimator: Improving iterative image decoding with gradient communication,” in *DCC*, 2020, pp. 23–32.
- [8] Ankur Mali, Alexander G. Ororbia, Dan Kifer, and C. Lee Giles, “An empirical analysis of recurrent learning algorithms in neural lossy image compression systems,” in *2021 Data Compression Conference (DCC)*, 2021, pp. 356–356.
- [9] Zhenyu Liu, Xianyu Yu, Yuan Gao, Shaolin Chen, Xiangyang Ji, and Dongsheng Wang, “Cu partition mode decision for HEVC hardwired intra encoder using convolution neural network,” *IEEE Transactions on Image Processing*, vol. 25, no. 11, pp. 5088–5103, 2016.
- [10] Yannick Strümpler, Ren Yang, and Radu Timofte, “Learning to improve image compression without changing the standard decoder,” *arXiv preprint arXiv:2009.12927*, 2020.
- [11] S. Takamura and M. Takagi, “Lossless image compression with lossy image using adaptive prediction and arithmetic coding,” in *DCC*, March 1994, pp. 166–174.
- [12] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli, “End-to-end optimized image compression,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.
- [13] George Toderici, Sean M. O’Malley, Sung Jin Hwang, Damien Vincent, David Minnen, Shumeet Baluja, Michele Covell, and Rahul Sukthankar, “Variable rate image compression with recurrent neural networks,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.
- [14] Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu, “Pixel recurrent neural networks,” in *ICML*, 2016, pp. 1747–1756.
- [15] Karol Gregor, Frederic Besse, Danilo Jimenez Rezende, Ivo Danihelka, and Daan Wierstra, “Towards conceptual compression,” in *NIPS*, pp. 3549–3557, 2016.
- [16] George Toderici, Sean M. O’Malley, Sung Jin Hwang, Damien Vincent, David Minnen, Shumeet Baluja, Michele Covell, and Rahul Sukthankar, “Variable rate image compression with recurrent neural networks,” *CoRR*, vol. abs/1511.06085, 2015.
- [17] George Toderici, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor, and Michele Covell, “Full resolution image compression with recurrent neural networks,” *CoRR*, vol. abs/1608.05148, 2016.
- [18] Nick Johnston, Damien Vincent, David Minnen, Michele Covell, Saurabh Singh, Troy Chinen, Sung Jin Hwang, Joel Shor, and George Toderici, “Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4385–4393.
- [19] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár, “Lossy image compression with compressive autoencoders,” *CoRR*, vol. abs/1703.00395, 2017.

- [20] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston, “Variational image compression with a scale hyperprior,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.
- [21] Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc V Gool, “Soft-to-hard vector quantization for end-to-end learning compressible representations,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 1141–1151.
- [22] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, “Energy compaction-based image compression using convolutional autoencoder,” *IEEE Transactions on Multimedia*, pp. 1–1, 2019.
- [23] S. Li, Z. Zheng, W. Dai, and H. Xiong, “Lossy image compression with filter bank based convolutional networks,” in *DCC*, March 2019, pp. 23–32.
- [24] Hossein Talebi, Damien Kelly, Xiyang Luo, Ignacio Garcia Dorado, Feng Yang, Peyman Milanfar, and Michael Elad, “Better compression with deep pre-editing,” *IEEE Transactions on Image Processing*, vol. 30, pp. 6673–6685, 2021.
- [25] Richard Shin, “JPEG-resistant adversarial images,” 2017.
- [26] Hossein Talebi, Damien Kelly, Xiyang Luo, Ignacio Garcia Dorado, Feng Yang, Peyman Milanfar, and Michael Elad, “Better compression with deep pre-editing,” 2020.
- [27] Karen Simonyan and Andrew Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [28] Ilya Sutskever, James Martens, and Geoffrey Hinton, “Generating text with recurrent neural networks,” Madison, WI, USA, 2011, ICML’11, p. 1017–1024, Omnipress.
- [29] Lukas Cavigelli, Pascal Hager, and Luca Benini, “CAS-CNN: A deep convolutional neural network for image compression artifact suppression,” *2017 International Joint Conference on Neural Networks (IJCNN)*, May 2017.
- [30] Z. Wang, E. P. Simoncelli, and A. C. Bovik, “Multiscale structural similarity for image quality assessment,” in *The Thirty-Seventh Asilomar Conference on Signals, Systems Computers, 2003*, 2003, vol. 2, pp. 1398–1402 Vol.2.
- [31] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang, “The unreasonable effectiveness of deep features as a perceptual metric,” 2018.
- [32] Sam Hasinoff, Dillon Sharlet, Ryan Geiss, Andrew Adams, Jonathan T. Barron, Florian Kainz, Jiawen Chen, and Marc Levoy, “Burst photography for high dynamic range and low-light imaging on mobile cameras,” *SIGGRAPH Asia*, 2016.
- [33] Eirikur Agustsson and Radu Timofte, “Ntire 2017 challenge on single image super-resolution: Dataset and study,” 07 2017, pp. 1122–1131.
- [34] Radu Timofte, Eirikur Agustsson, Luc Van Gool, Ming-Hsuan Yang, Lei Zhang, Bee Lim, et al., “Ntire 2017 challenge on single image super-resolution: Methods and results,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.
- [35] Kede Ma, Qingbo Wu, Zhou Wang, Zhengfang Duanmu, Hongwei Yong, Hongliang Li, and Lei Zhang, “Group MAD competition? A new methodology to compare objective image quality models,” in *CVPR*, 2016, pp. 1664–1673.
- [36] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE Trans. Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.