

Sandwiched Compression: Repurposing Standard Codecs with Neural Network Wrappers

Onur G. Guleryuz, Philip A. Chou, Berivan Isik, Hugues Hoppe, Danhang Tang,
Ruofei Du, Jonathan Taylor, Philip Davidson, and Sean Fanello

Abstract—We propose sandwiching standard image and video codecs between pre- and post-processing neural networks. The networks are jointly trained through a differentiable codec proxy to minimize a given rate-distortion loss. This sandwich architecture not only improves the standard codec’s performance on its intended content, it can effectively adapt the codec to other types of image/video content and to other distortion measures. Essentially, the sandwich learns to transmit “neural code images” that optimize overall rate-distortion performance even when the overall problem is well outside the scope of the codec’s design. Through a variety of examples, we apply the sandwich architecture to sources with different numbers of channels, higher resolution, higher dynamic range, and perceptual distortion measures. The results demonstrate substantial improvements (up to 9 dB gains or up to 30% bitrate reductions) compared to alternative adaptations. We derive VQ equivalents for the sandwich, establish optimality properties, and design differentiable codec proxies approximating current standard codecs. We further analyze model complexity, visual quality under perceptual metrics, as well as sandwich configurations that offer interesting potentials in image/video compression and streaming.

Index Terms—Image/video/graphics coding, learned compression, differentiable proxy, rate-distortion optimization, multi-spectral, super-resolution, high dynamic range, perceptual distortion measures

I. INTRODUCTION

Image and video coding are well-established domains, with a rich history marked by the evolution of international and industry standard codecs, such as JPEG, MPEG 1,2,4, H261, H262, H264/AVC, VC1, VP9, H265/HEVC, AV1, and H266/VVC [1]–[6]. Remarkably, for several decades, these standard codecs have been fundamentally rooted in linear transforms like the discrete cosine transform (DCT) in the spatial dimensions, and motion-compensated prediction in the temporal dimension. At this time, however, image and video coding is beginning to shift in a new direction.

Recent advancements have spotlighted *learned* image and video codecs based on neural networks trained end-to-end that, impressively, outperform the standard codecs in rate-distortion metrics, where distortion is measured as mean squared error (MSE) on color image components [7]–[13]. In scenarios where complete faith to the source is not demanded, more dramatic reductions in bitrate ($\sim 3\times$) for the same visual quality have been obtained by training neural networks using auxiliary distortion measures (such as image likelihood modeled by discriminators in generative adversarial networks [14]). Neural networks also have an obvious, if unheralded, functional advantage over standard codecs in that they can be trained on

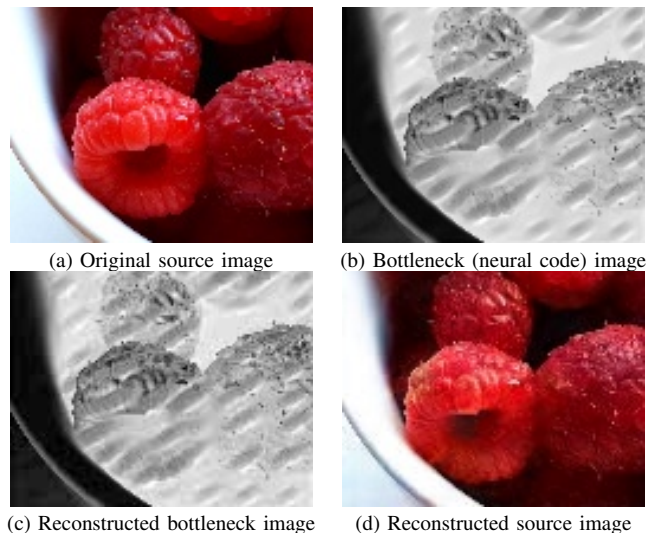


Fig. 1. The sandwich architecture can accomplish surprising results even with a simple codec (here JPEG 4:0:0, a single-channel grayscale codec). The neural pre-processor is able to encode the full RGB image in (a) into a grayscale image of neural codes in (b). The neural codes are low-frequency dither-like patterns that modulate the color information yet also survive JPEG compression (c). At the decoding end, the neural post-processor demodulates the patterns to faithfully recover the color while also achieving deblocking.

datasets of images whose distributions are mismatched from the usual photographic images, for example medical, multi-spectral, depth, geometric, or other unusual image classes, as well as other distortion criteria, including human perceptual criteria but also machine performance criteria (e.g., classification, segmentation, labeling, and diagnosis.)

Unfortunately, the performance and functional advantages of neural codecs have so far come at great computational cost [8], [10]–[13]. This cost is typically at a level that is impractical for HD imagery at video rates even on dedicated neural chips especially in mobile devices where power is a prime concern. Handling UHD at graphics rates is even more impractical.

In this paper, we propose an architecture for a *hybrid* between standard codecs and purely neural codecs, which we call the *sandwich architecture*. In the sandwich architecture, a standard codec is positioned between a neural pre-processor and a neural post-processor, which are jointly trained to minimize distortion subject to a bitrate constraint. The neural pre- and post-processors can be lightweight, yet they are able to improve the rate-distortion performance of the standard codec, even on typical color photographic images when the distortion measure is MSE — the design target for standard codecs. More interestingly, the improvement is especially pronounced

when the application mismatches the design target in some way, including non-RGB images (e.g., C -channel images with $C \neq 3$ such as medical, multispectral, depth, geometric, and other sensed images), non-MSE distortion criteria (e.g., human perceptual metrics, realism metrics, and machine task-specific performance metrics), and non-standard profile hardware constraints (e.g., higher bit depth and higher spatial resolution). At the same time, the sandwich approach leverages the standard codec for much of the heavy lifting, including highly efficient transforms, entropy coding, and motion processing. Vast resources have been put into the hardware implementations of standard codecs and their broader ecosystems (transparent packetization, networking, routing, etc.), which can make these resources essentially free compared to the power consumption required in neural chips.

The magic behind the sandwich architecture is the ability for the pre-processor to learn how to produce images of *neural codes* that are well-compressible by a standard codec and for the post-processor to learn how to decode these images, to minimize the relevant distortion measure. Of course, the neural code images have to be robust to the compression noise typically inserted at those bitrates by the standard codec. To gain an intuitive understanding of what these neural code images may look like, consider a simplified problem in which the neural pre- and post-processors adapt a 1-channel (grayscale) codec to compressing an ordinary 3-channel color image, shown in Fig. 1. The top image (a) is the *original source image* fed into the pre-processor. The neural code image (b), or *bottleneck image*, is the image produced by the pre-processor and fed to into the encoder of the standard codec. This is called the bottleneck image since it is at the locus of the compression bottleneck. Note that the bottleneck image contains spatial modulation patterns (akin to watermarks) that serve to encode the color information in this case. These patterns are the neural codes. Here the neural codes have a single channel, but more typically they have more channels. The *reconstructed bottleneck image* (c) is emitted from decoder of the standard codec (note the typical JPEG blocking artifacts) and fed into the post-processor. The *reconstructed source image* (d) is the image output from the post-processor. Note that both color and sharp spatial definition are recovered from the neural codes. (Refer to subsection IV-A for rate-distortion results relevant to this scenario.)

This journal paper aims to be a coherent presentation of our recent work in this area [15]–[18]. Few works prior to ours paired standard codecs with neural networks as pre- or post-processors. Most such works paired the standard codec with either a neural pre-processor alone (e.g., to perform denoising of the input image [19]–[21]) or a neural post-processor alone (e.g., to perform deblocking or other enhancements of the output image [22]–[24]). A few works paired standard codecs with both neural pre- and post-processors, such as [25]–[29], but these solutions, like prior non-neural solutions such as [30] or the “frame super-resolution” mode of VP9, did so in such a way that the pre- and post-processors may be used independently; thus no neural codes are generated; hence they do not take full advantage of the communication available between the pre- and post-processors (see Proposition 1).

Beyond proposing the sandwich architecture itself, a major contribution of our paper is a solution for *jointly training the pre- and post-processors*. To jointly train these neural networks using standard gradient back-propagation, the standard codec must be differentiable. Hence during training we replace the standard codec with a differentiable *codec proxy*. We show that well-designed simple proxies that approximate key codec components allow the training of models that robustly work with different standard codecs. In so doing, and with help of many scenario results we show, we also establish the sandwich as a potent avenue for compression problems with complexity/legacy constraints.¹

Another major contribution of this work is to demonstrate the *advantages of the sandwich architecture across a variety of settings*:

- For coding of 3-channel color images over a 1-channel (grayscale, or 4:0:0) codec, as in Fig. 1, sandwiching has 6–9 dB gain in MSE. Over a 1.5-channel (4:2:0) codec, sandwiching has a 10% reduction in bitrate. And over a 3-channel (4:4:4) codec, sandwiching has a 15% reduction in bitrate.
- For coding of 2x high resolution (HR) or super-resolved images over 1x lower resolution (LR) codecs, sandwiching has up to 9 dB gain in MSE.
- For coding of 16-bit high dynamic range² (HDR) color images over an 8-bit lower dynamic range (LDR) codec, sandwiching has up to 3 dB gain in MSE.
- In rendering applications, for coding of 3-channel normal maps over a 1.5-channel codec, sandwiching has a 4–5 dB gain in MSE. Over a 3-channel codec, sandwiching has a 1.5–2 dB gain, or about 15% reduction in bitrate.
- For coding of 8-channel texture maps (3-channel albedo, 3-channel normals, 1-channel roughness, and 1-channel occlusion) over an 8-channel codec (implemented as the concatenation of eight 1-channel instances), sandwiching has a 20–30% reduction in bitrate, when distortion is measured in terms of MSE of the lighted, rendered images (the so-called *shaded distortion*).
- We show that the results do not depend heavily on whether the standard codec is JPEG or HEVC-Intra (HEIC) without any model retraining. We also show that the results degrade but hold up well as the number of parameters of the pre- and post-processors is reduced by more than two orders of magnitude, to fewer than 60K parameters in each neural processor.
- For coding of video, we show analogous gains in MSE for coding color over grayscale codecs, and coding HR over LR codecs. Perhaps most importantly from the perspective of video applications, we demonstrate that for coding color video over color codecs, sandwiching yields 30% reduction in bitrate at the same visual quality, when trained to minimize the perceptual distortion measure LPIPS instead of MSE.

The rest of the paper is organized as follows. After the prelude of Section II, which discusses VQ equivalents and optimality, Section III presents the sandwich architecture. Section IV includes our image compression experiments followed

¹As will become clear one can immediately benefit many legacy scenarios by deploying a sandwich with constraint-dependent models trained with this paper’s proxies.

²In this work, the term HDR is interchangeable with high bit depth.

by Section V which presents complexity results. Section VI is devoted to video compression experiments. Section VII is a discussion and conclusion.

II. PRELUDE: THE SANDWICH AS A CODELENGTH CONSTRAINED VECTOR QUANTIZER

Pre-Post processing applied around a compression codec is a well-known technique. In $\Sigma\Delta$ compression [31] one wraps a simple 1-bit quantizer to make it function like a k -bit one, in [32] one wraps DPCM codecs (performance-wise inferior to transform codecs) to get them to perform like transform codecs, using [19]–[24] one can wrap image codecs to reduce input noise and reduce codec artifacts, and so on. Compression literature includes many such interesting designs that offer specific solutions to specific problems. With neural networks one now has the capability of designing much more general mappings as pre-post processors. In this section we briefly explore the potential gains one can tap into.

Compression codecs can be seen as vector quantizer codebooks. A standard codec at a particular operating point can be thought of in terms of a set of codewords (decoder reconstruction vectors existing in high dimensions) and associated binary strings (bits signaling each desired reconstruction). A sandwich with non-identity wrappers maps a source to use the standard codebook and then maps the standard decoder’s output into final reconstructions. Looking from outside the sandwich, we hence see a new codebook for the source that is determined by the pre/post-processor mappings modifying the standard codebook. Suppose the standard codec is not adequate for a given source. Then, a natural question to ask is “*How much better can we make the standard codec by wrapping it in a sandwich?*”

In order to quantify the properties of “sandwich-achievable” codebooks and how they would compare to a codebook that is optimal for the source, let us momentarily disregard limitations on neural network complexity and limitations of back propagation in finding overall optimal solutions. Assume we can find the optimal pre-post-processor mappings. What is the efficiency of the sandwich system with respect to an optimal codebook? The following proposition shows that the optimal sandwich can accomplish the optimal compression performance except for a potential rate penalty induced by a potential mismatch of the standard codec codelengths.

Proposition 1. *Let X be a \mathbb{R}^n -valued bounded source, let d be a distortion measure, and let $D(R)$ be the operational distortion-rate function for X under d . For any $\epsilon > 0$, let $(\alpha^*, \beta^*, \gamma^*)$ be the encoding, decoding, and lossless coding maps for a rate- R codec for X achieving $D(R)$ within $\epsilon/2$. Let (α, β, γ) be a regular codec (e.g., a standard codec, possibly designed for a different source and different distortion measure) with bounded codelengths. Then there exist neural pre- and post-processors f and g such that the codec sandwich $(\alpha \circ f, g \circ \beta, \gamma)$ has expected distortion at most $D(R) + \epsilon$ and expected rate at most $R + D(p||q) + \epsilon$, where $p(k) = P\{\alpha^*(X) = k\}$ and $q(k) = 2^{-\lceil \gamma(k) \rceil}$.*

Proof. See subsection VIII-H in Supplementary Material. \square

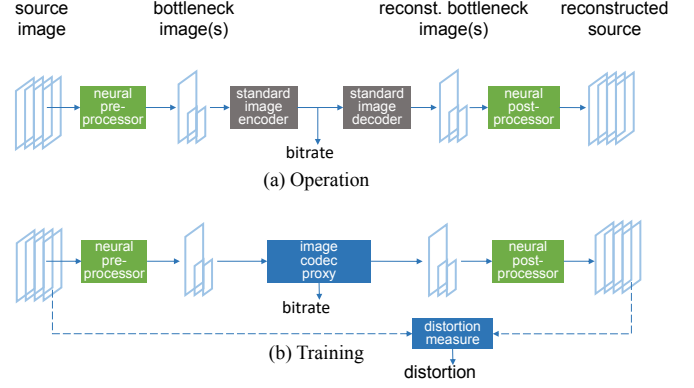


Fig. 2. Neural-sandwiched image codec during (a) operation and (b) training. Gray boxes are not differentiable; blue are differentiable; green are trainable. Loss function for training is $\sum D_n + \lambda R_n$ over example images n .

Note the key role of the sandwich in adapting the inner codebook to the outer compression scenario. When sandwiching a high-performance image/video codec for different but related image/video applications one can expect the mismatch to be lighter compared to, say, when one tries to sandwich an image codec to transport audio data. From the perspective of the proposition, using configurable codecs, *i.e.*, those that allow codebook codelengths to be optimized such as neural codecs, may help minimize the implied penalty. While beyond the scope of this paper, we point to generalizing the sandwich to configurable codecs as an interesting research direction.

III. THE SANDWICH ARCHITECTURE

In this section, we set up the sandwich architecture. We focus on image and video compression in turn.

A. Sandwich for Image Compression

The sandwich architecture for image compression is shown in Fig. 2(a). An *original source image* S with one or more full-resolution channels is mapped by a neural pre-processor into one or more channels of *neural* (or *latent*) *codes*. Each channel of neural codes may be full resolution or reduced resolution. The channels of neural 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 post-processor 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, or one full-resolution channel as an image in 4:0:0 (*i.e.*, grayscale) format — all without color conversion. Other combinations of channels are processed by appropriate grouping.

Fig. 3 shows the network architectures we use for our neural pre-processor and post-processor. The upper branch of the network learns pointwise operations, like color conversion,

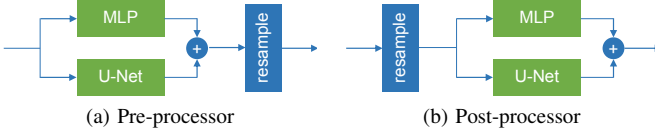


Fig. 3. Neural pre-processor and post-processor.

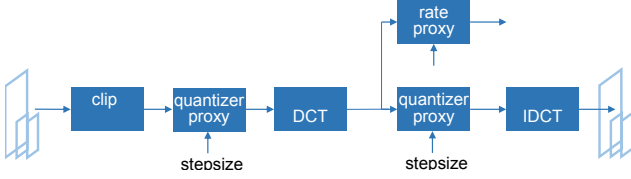


Fig. 4. Image codec proxy.

using a multilayer perceptron (MLP) or equivalently a series of 1×1 2D convolutional layers, while the lower branch uses a U-Net [33] to take into account more complex spatial context. At the output of the pre-processor, any half-resolution channels are obtained by sub-sampling, while at the input of the post-processor, any half-resolution channels are first upsampled to full resolution. We have deliberately picked the U-Net as it is a well-known model whose performance in various areas is well-documented. U-Net models also have been systematically studied with reduced parameter/complexity variants easily generated.

Fig. 2(b) shows the setup for training the neural pre-processor and post-processor using stochastic 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 $n = 1, \dots, N$, 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 any differentiable distortion measure $D_n = d(S_n, \hat{S}_n)$ (such as the squared ℓ_2 error $\|S_n - \hat{S}_n\|^2$) 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 pre-processor and post-processor 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 pre-processor and post-processor to produce bottleneck images sufficiently like natural images that they can also be compressed efficiently by other codecs such as HEVC. The image codec proxy spatially partitions the input channels into 8×8 blocks. In the DCT domain, the blocks $X = [X_i]$ are processed independently, using (1) a “differentiable quantizer” (or *quantizer proxy*) to create distorted DCT coefficients $\hat{X}_i = Q(X_i)$, and (2) a differentiable entropy measure (or *rate proxy*) to estimate the bitrate required to represent the distorted coefficients \hat{X}_i . Both proxies take the

³In this paper as in most of the ML literature, the term *differentiable* more properly means *almost-everywhere differentiable*.

nominal quantization stepsize Δ as an additional input.

Further information on quantizer and rate proxies is provided in supplementary section VIII-A and information on adaptations for HR and HDR are provided in VIII-B.

B. Sandwich for Video Compression

The sandwich architecture for video compression is shown wrapping our video codec proxy in Fig. 5. Observe that the neural pre-post-processors handle video frames independently. The video codec proxy maps an input video sequence to an output video sequence plus a bitrate for each video frame. It has the following components.

Intra-Frame Coding. The video codec proxy simulates coding the first ($t = 0$) frame of the group, or the I-frame, using the image codec proxy described above in subsection III-A.

Motion Compensation. The video codec proxy simulates predicting each subsequent ($t > 0$) frame of the group, or P-frame, by motion-compensating the previous frame. Motion compensation is performed using a pre-computed dense motion flow field obtained by running a state-of-the-art optical flow estimator, UFlow [34], between the *original* source images $S_n(t)$ and $S_n(t-1)$. The video proxy simply applies this motion flow to the previous reconstructed bottleneck image $\hat{B}_n(t-1)$ to obtain an inter-frame prediction $\hat{B}_n(t)$ for bottleneck image $B_n(t)$. Note that our motion compensation proxy does not actually depend on $B_n(t)$, so even though it is a spatial warping, it is a linear map from $\hat{B}_n(t-1)$ to $\hat{B}_n(t)$, with a constant Jacobian. This makes optimization much easier than using a differentiable function of both $\hat{B}_n(t-1)$ and $B_n(t)$, like UFlow itself, that *finds* as well as *applies* a warping from $\hat{B}_n(t-1)$ to $B_n(t)$. Such functions have notoriously fluctuating Jacobians that make training difficult.

Prediction Mode Selection. An Intra/Inter prediction proxy simulates a modern video codec’s Inter/Intra prediction mode decisions. This ensures better handling of temporally occluded/uncovered regions in video. First, Intra prediction is simulated by rudimentarily compressing the current-frame and low pass filtering it. This simulates filtering, albeit not the usual directional filtering, to predict each block from its neighboring blocks. Initial rudimentary compression ensures that the Intra prediction proxy is not unduly preferred at very low rates. For each block, the Intra prediction (from spatial filtering) is compared to the Inter prediction (from motion compensation), and the one closest to the input block determines the mode of the prediction.

Residual Coding. The predicted image, comprising a combination of Intra- and Inter-predicted blocks, is subtracted from the bottleneck image, to form a prediction residual. The residual image is then compressed using the image codec proxy described above in subsection III-A. The compressed residual is added back to the prediction to obtain a “pre-loop-filtered” reconstruction of the bottleneck image $\hat{B}_n(t)$.

Loop Filtering. The “pre-loop-filtered” reconstruction is then filtered by a loop filter to obtain the final reconstructed bottleneck image $\hat{B}_n(t)$. The loop filter is implemented with a small U-Net((8);(8, 8)) [33] (see Section V for U-Net notation) that processes one channel at a time. The loop filter is trained

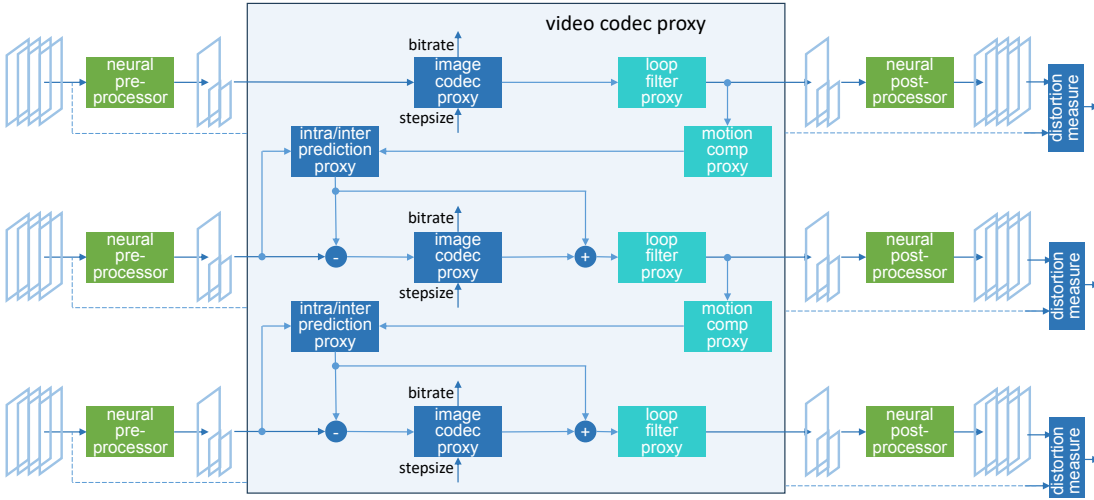


Fig. 5. Neural-sandwiched video codec during training. Loss function for training is $\sum D_n(t) + \lambda R_n(t)$ over example clips n and frames t . The shaded box (video codec proxy) is replaced with a standard video codec during operation/inference. Green boxes are trainable; blue are differentiable; cyan are differentiable with pre-trained parameters. The entire video codec proxy is differentiable.

once for four rate points on natural video using only the video codec proxy with rate-distortion (ℓ_2) training loss in order to mimic common loop filters. The resulting set of filters are kept fixed for all of our simulations, *i.e.*, once independently trained, the four loop filters are not further trained.

Pre-Post-Processors. Same as [subsection III-A](#) and [Fig. 3](#).

Loss Function. The loss function is the total rate-distortion Lagrangian $\sum_{n,t} D_n(t) + \lambda R_n(t)$, where $D_n(t)$ and $R_n(t)$ are the distortion and rate of frame t in clip n . The rate term in particular serves to encourage the pre- and post-processors to produce temporally consistent neural codes, since neural codes that move according to the motion field are well predicted (see [subsection VI-B](#).) Note that the overall mapping from the input images through the pre-processor, video codec proxy, post-processor, and loss function is differentiable.

IV. IMAGE COMPRESSION EXPERIMENTS

We first present results for compressing ordinary 3-channel color (RGB) images through codecs with a restricted number of channels (4:0:0 and 4:2:0 compared to 4:4:4), where distortion is measured as RGB-PSNR. Then we present results for compressing high spatial resolution (HR) images through codecs with lower spatial resolution (LR), and for compressing high dynamic range (HDR) images through codecs with lower dynamic range (LDR), where distortion is again measured as RGB-PSNR. These results are indicative of how a neural sandwich can adapt the hardware of a standard codec to source images with higher resource requirements. Finally, we present results on graphics data, first for compressing 3-channel normal maps, where distortion is measured as PSNR on the normal maps, and then for compressing 8-channel shading maps for use in computer graphics, where distortion is measured as RGB-PSNR after shading from 8 to 3 channels. These results are indicative of how a neural sandwich can adapt a codec designed for 3-channel RGB images and MSE to other image types and other distortion measures. In these results, we also see how the codec proxy, though modeled after

JPEG, is also adequate to represent more advanced codecs such as HEIC/HEVC.

For our RGB results, we use the Pets, CLIC, and HDR+ datasets [\[35\]–\[37\]](#), while for our computer graphics results, we use the Relightables dataset [\[38\]](#). Training and evaluation are performed on distinct subsets of each dataset. All results are generated using actual compression on 500 test images of size 256×256 randomly cropped from the evaluation subset.

The U-Nets have a multi-resolution ladder of four with channels doubling up the ladder from 32 to 512, specifically U-Net([32, 64, 128, 256]; [512, 256, 128, 64, 32]). (See [Section V](#) for notation.) MLP networks have two layers with 16 nodes. Output channels of the networks are determined based on bottleneck and overall output channels.

We obtain R-D curves as follows. We train four models m_i for four different Lagrange multiplier values λ_i using established $D + \lambda R$ optimization [\[39\]](#). 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.

A. Compressing 3-channel RGB Images with $C \leq 3$ -channel Codecs

In this section, we report the rate-distortion performance of compressing 3-channel RGB images with JPEG YUV and neural-sandwiched JPEG codecs, across 4:4:4, 4:2:0, and 4:0:0 formats, which respectively correspond to C -channel bottleneck images with $C = 3, 1.5, \text{ and } 1$.

[Fig. 6](#) shows R-D results evaluated on the Pets dataset. For the 4:0:0 format, the neural-sandwiched version shows 6–9 dB improvement over the standard codec, due to the fact that for this format, the standard codec can transport only grayscale, whereas the neural-sandwiched version manages to transport color through modulating patterns, as exemplified in [Fig. 1](#) (and [Fig. S2](#)). For the 4:2:0 and 4:4:4 formats, R-D performances for the standard codec are close to one another. For both formats, the neural-sandwiched standard codec performs

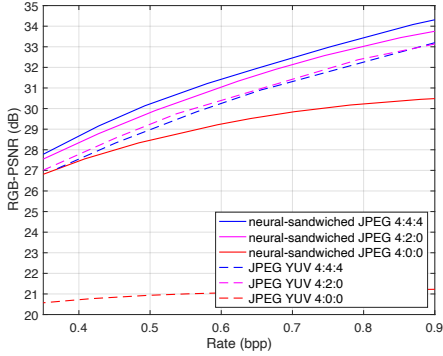


Fig. 6. R-D performance of compressing 3-channel RGB images with JPEG YUV and neural-sandwiched JPEG, in 4:4:4, 4:2:0, and 4:0:0.

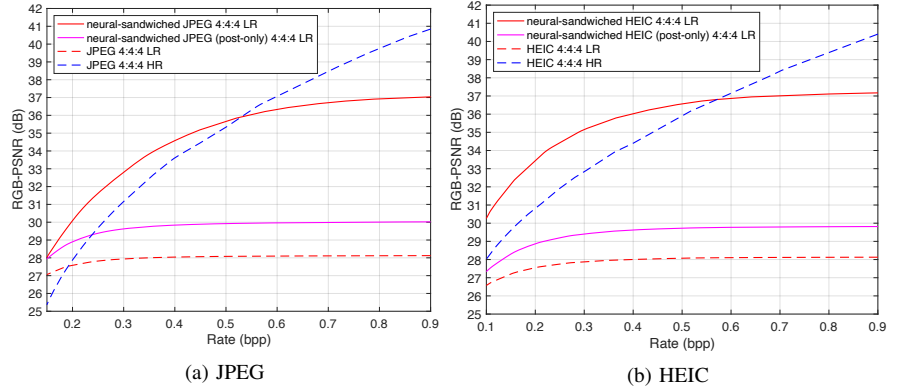


Fig. 7. R-D performances of compressing high resolution (HR) RGB images with lower resolution (LR) codecs alone, LR codecs plus neural post-processing, and neural-sandwiched SR codecs. For reference, also shown are R-D performances of compressing HR RGB images with HR codecs.

better than the standard codec in either format. In particular, for the 4:2:0 format, the neural-sandwiched version shows a 10% reduction in bitrate, while for the 4:4:4 format, the neural-sandwiched version shows a 15% reduction. It is remarkable that for the 4:4:4 format, the neural-sandwiched version finds a way to utilize the denser sampling for improved gains. Also surprising is that at low rates the neural-sandwiched 4:0:0 codec is competitive with the JPEG YUV codecs.

B. Compressing High Resolution (HR) RGB Images with Lower Resolution (LR) Codecs

In this subsection, we consider the rate-distortion performance of compressing high resolution (HR) RGB images with lower resolution (LR) standard codecs, both with JPEG and HEIC⁴ LR is half the horizontal and vertical resolution of HR. Thus, whether sandwiched or not, we precede the standard codecs with bicubic filtering and $2\times$ downsampling and follow them with $2\times$ upsampling using Lanczos3 interpolation.

Fig. 7 shows R-D results using (a) JPEG and (b) HEIC as the standard codec evaluated on the CLIC dataset. When they are neural-sandwiched, JPEG and HEIC have identical pre- and post-processors, with no retraining for HEIC. The neural-sandwiched standard codecs show substantial improvement over the standard codecs alone: 5-7 dB gains for JPEG, and 7-8 dB gains for HEIC, in the 0.3-0.5 bpp range, and further gains at higher rates. We also compare a standard codec with a post-processor alone (*i.e.*, with no pre-processor), where the post-processor is architecturally identical to the post-processor in the neural sandwich, but trained to perform only super-resolution. It can be seen that the post-processor alone accounts for at most 2 dB of the sandwich’s gains. The substantial improvement obtained by the sandwich over the super-resolution network clearly points to the importance of the neural pre-processor, the joint training of the pre- and post-processor networks, and their ability to communicate with each other using neural codes to signal *how* to super-resolve the images. For reference, the figure also shows R-D performances of compressing HR RGB images with codecs that are natively

⁴Our use of the term HEIC is a slight misnomer. We used x265/ffmpeg to generate HEVC-INTRA compressed images. While we have tried to minimize it, our per-image results thus contain some unneeded meta data bits.

bpp	0.2	0.3	0.4	0.5	0.6	0.7	0.8
CNN-RD [26]	1.58	1.09	0.67	0.55	0.33	0.18	0.15
HR sandwich	1.59	1.49	1.42	1.49	1.49	1.46	1.69

TABLE I

GAIN IN PSNR-Y (dB) OVER JPEG ON DIV2K VALIDATION IMAGE 0873.

HR. It can be seen that the sandwiched LR codecs outperform even the native HR codecs over a wide range of lower bitrates. Comparing JPEG and HEIC results, it can be seen that the gains due to neural sandwiching of JPEG are substantially retained for HEIC.

Fig. 8 (and Fig. S3) show qualitative and objective results. Observe the substantial improvements obtained by the sandwiched codec over JPEG and neural post-processing: Detail is retained in the city view, aliasing is avoided on the building face and the texture. All with substantial dB improvements (+4.5 dB, +6.5 dB, +10.6 dB over neural post-processing) at the same rate. Fig. 9 (and Fig. S4) show the sandwich bottlenecks.

Table I compares our HR sandwich to the closely related but independently developed work of [26], in which their “CNN-RD” solution also surrounds a standard codec with neural pre- and post-processors using $2\times$ down- and up-sampling. However their networks’ formulation and training regimen prohibits them from learning to communicate the neural codes needed to carry good HR information. (Their post-processor is trained first to super-resolve a low-pass image; then their pre-processor is trained to minimize $D + \lambda R$ with the fixed post-processor. This misses the main advantage of having neural pre- and post-processors.) The table shows that our work has significantly higher gains in PSNR-Y (dB) relative to the same standard codec (JPEG) on the Div2k validation image 0873 [40]. Indeed, though not shown in the table, their solution saturates and begins to under-perform the standard codec above 0.8 bpp (~ 30 dB); ours out-performs the standard codec until about 2.0 bpp (~ 37 dB).

C. Compressing HDR RGB Images with LDR Codecs

In this subsection, we report the rate-distortion performance of compressing high dynamic range (HDR) RGB images with lower dynamic range (LDR) standard codecs (8-bit JPEG and 8-bit HEIC), alone and neural-sandwiched.

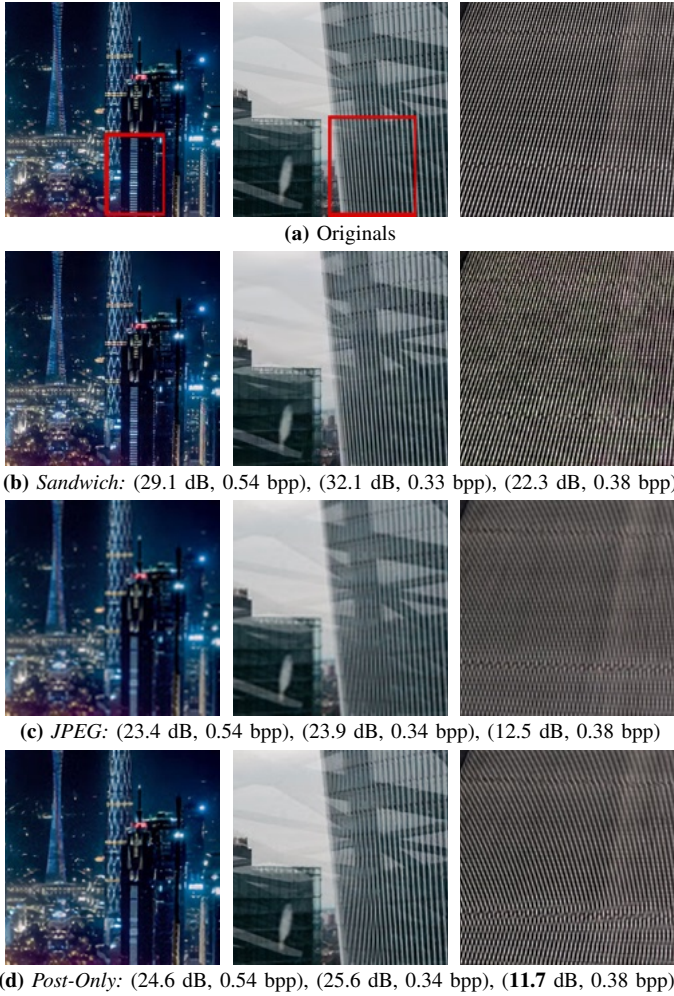


Fig. 8. Super-resolution sandwich of a low-res codec: Original 256×256 source images and reconstructions by sandwich, JPEG with linear upsampling, and JPEG with neural post-processing respectively. The regions identified in the top row show areas where detail is either lost or aliased after downsampling and LR transport. Note how the sandwich output in (b) correctly transports the detail whereas JPEG and post-only recover the wrong information. The picture in the last column, while correctly transported by the sandwich, results in severe aliasing for JPEG and even further reduced performance for post-only which amplifies the aliasing.

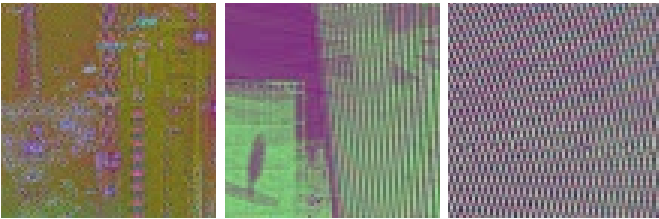


Fig. 9. 128×128 reconstructed bottleneck images for the super-resolution sandwich results in Fig. 8 [enlarged for clarity]. Observe that while the bottlenecks appear aliased, noisy etc., the sandwich post-processor has correctly demodulated this noise in the final pictures.

For the HDR simulations, we use the HDR+ dataset [37]. The HDR images have $d = 16$ bits per color component, while the LDR codecs transmit only 8 bits per color component. By sandwiching the LDR codecs in neural pre- and post-processors, it is possible to signal spatially-localized tone mapping curves via neural codes from the pre-processor to the post-processor, in order to carry the least significant bits

of the HDR image through the LDR codec.

Fig. 10 shows R-D results in terms of d -bit RGB-PSNR vs. bits per pixel, evaluated on the HDR+ dataset. The definition of d -bit RGB-PSNR is

$$\text{RGB PSNR} = 10 \log_{10} \left((2^d - 1)^2 (3HW) / \|S - \hat{S}\|^2 \right), \quad (1)$$

where S and \hat{S} are the original and reproduced d -bit RGB source images of size $H \times W$. The figure shows the performance of the LDR codecs alone (8-bit JPEG and 8-bit HEIC) in comparison to the neural-sandwiched LDR codecs, as well as to JPEG post-processed with the state-of-the-art Dequantization-Net [41] (trained on the same dataset). The maximum PSNR one can obtain by losslessly encoding the most significant 8-bits is illustrated as LDR saturation. The standard codecs alone, or with the Dequantization-Net post-processor only, saturate at that level. Observe that the sandwiched codecs rise up to 3 dB above the saturation line, highlighting the importance of joint training of the pre- and post-processors and communication between them using neural codes. Unfortunately the software implementing the standard codecs precludes the transmission of higher rates. Neither our JPEG nor HEIC implementation is able to go beyond ~ 3 bpp on average. For all R-D curves the highest rate point is where the software cuts off. Using codec implementations accomplishing higher rates, the gains of the sandwich are expected to increase further.

D. Compressing 3-channel Normal Maps with Color Codecs

In this subsection, we report the rate-distortion performance of compressing 3-channel normal maps with standard (color) codecs, both alone and neural-sandwiched. An example normal map is shown in Fig. 11. Normal maps in computer graphics are used to increase the perceived resolution of a mesh [42], [43]. 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. Note that three channels are redundant since $n_x^2 + n_y^2 + n_z^2 = 1$. To represent these as 8-bit images, we map each channel $[-1, 1]$ to RGB $[0, 255]$.

Fig. 12 (and Fig. S5) show R-D results using HEIC (and JPEG) as the standard codec, in terms of PSNR vs bits per pixel, evaluated on normal maps from the Relightables dataset [38]. The best results are obtained with the neural-sandwiched 4:4:4 codecs, which exploit the channel redundancy. Neural-sandwiching codecs with 4:2:0 and 4:0:0 formats provide significant gains over their respective baselines. YUV 4:2:0 and YUV 4:4:4 codecs use RGB-YUV conversion, which does not provide any advantage for this dataset. Codecs with 4:4:4 (no color conv.) are better. As before, the neural-sandwiched codecs do not use color conversion. The best non-neural result (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-sandwiched codec with a 4:4:4 format. Comparing JPEG and HEIC results, it can be seen (again) that the gains due to neural sandwiching of JPEG are substantially retained for HEIC, with about 15% reduction in bitrate compared to HEIC alone.

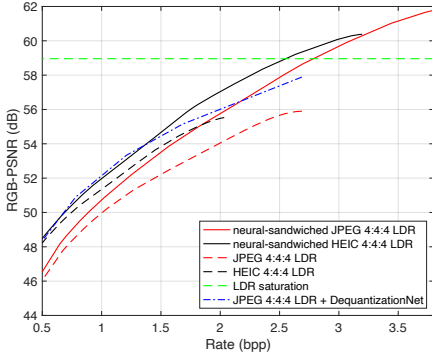


Fig. 10. R-D performances of compressing HDR RGB images with LDR codecs, neural-sandwiched LDR codecs, and LDR codecs post-processed with Dequantization-Net [41]. LDR saturate indicates PSNR of rounding d bits to 8 and lossless coding.

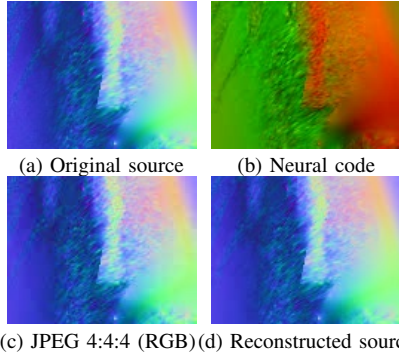


Fig. 11. Compression of a normal-map image at 0.8 bits/pixel. JPEG 4:4:4 (RGB) achieves 32.2 dB, whereas the neural-sandwiched 4:4:4 format attains 34.2 dB. Refer to Fig. S5 for full R-D results.

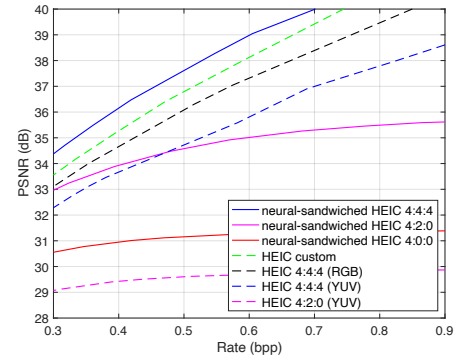


Fig. 12. R-D performances of compressing normal map images with HEIC and neural-sandwiched HEIC, in various formats.

E. Compressing 8-channel Texture Maps with 3-channel Color Codecs and Shaded Distortion

A common technique in computer graphics is to render a surface using *texture mapping*, which stores sampled surface properties in an associated texture atlas image (see Fig. 13). The texture map often contains not just albedo (RGB color) but additional surface properties (e.g., surface normals, roughness, ambient occlusion) that enable more realistic shading. In this section, we report rate-distortion performance of compressing 8-channel texture maps with and without sandwiching.

We briefly review the rendering process. To render a surface mesh from a particular view and with particular lighting, rasterization identifies the screen-space pixels covered by the mesh triangles. For each pixel, it obtains the interpolated 3D surface position as well as interpolated texture coordinates. Then, a *pixel shader* computes the view direction, the light direction(s), and the sampled texture attributes for the surface point. The final shaded RGB color for the pixel is a complicated formula involving all of these inputs.

For compression in this scenario, it is natural to measure distortion not over the texture image values but over the final rendered pixel colors. Specifically, we measure the average RGB MSE of images rendered using a collection of typical views and lighting conditions. We call this *shaded distortion*.

In principle, it should be possible to train the neural sandwich end-to-end by measuring shaded distortion over rendered images using a *differentiable renderer*. However, this proves challenging for the large texture map sizes (up to 4K) encountered in practice.

Instead, our approach during training is to measure shaded distortion in the domain of the texture images. That is, we compute shading using all the 3D parameters of the traditional rendering, but output the resulting shaded colors onto an image defined over the texture domain itself (see Fig. 13e). The key benefit is that the computation is local, so training can use cropped texture images. (For final evaluation, we measure shaded distortion over traditionally rendered images.)

In our experiments, we use a texture map with $C = 8$ channels: 3 RGB albedo channels, 3 normal map channels (as in the previous subsection), 1 roughness channel, and 1

occlusion channel, as illustrated in Fig. 13. We refer to this as a $[3, 3, 1, 1]$ -channel texture map.

Fig. 14 shows R-D results in terms of PSNR of the shaded distortion in dB vs bits per texture map pixel. Compression with and without neural sandwiching are compared. Without sandwiching, to compress the 8-channel texture map with a standard codec, we partition the texture map into its natural components, here with $[3, 3, 1, 1]$ channels, and compress each component separately (with one codec using 4:4:4 format and color conversion, one codec using 4:4:4 format and no color conversion, and two codecs using 4:0:0 format). With sandwiching, to compress the 8-channel texture map, for concreteness we choose 8-channel bottleneck images at the same resolution as the texture map, and code each of the 8 bottleneck channels as a grayscale image using HEIC 4:0:0. It can be seen from the figure that neural-sandwiching provides a 20-30% reduction in bitrate compared to HEIC alone. This example illustrates that neural sandwiching can be used for non-RGB images with $C > 3$ channels and non-standard distortion measures.

V. COMPLEXITY EXPERIMENTS

In the image compression experiments of Section IV, we use an MLP in parallel with a U-Net. The MLP is relatively simple, with two hidden layers, each with 16 hidden channels. Most of the complexity is in the U-Net, which in its standard form [33] has four 2-layer 3×3 convolutional “encoder” blocks each followed by a $2 \times$ downsampling, followed by five 2-layer 3×3 convolutional “decoder” blocks separated by $2 \times$ upsampling and concatenation with the output of the same-resolution encoder block. The number of channels output from the encoder blocks is $[32, 64, 128, 256]$, while the number of channels output from the decoder blocks is $[512, 256, 128, 64, 32]$, denoted U-Net($[32, 64, 128, 256]$; $[512, 256, 128, 64, 32]$). A final 3×3 convolutional layer produces C_{out} output channels. These tuples, along with C_{in} and C_{out} , can be used as hyper-parameters to specify the U-Net.

In this section, we perform a sweep over the U-Nets hyper-parameters to study the tradeoff between the complexity of the sandwich and its rate-distortion performance. For this study, we choose a particular application: compressing HR RGB

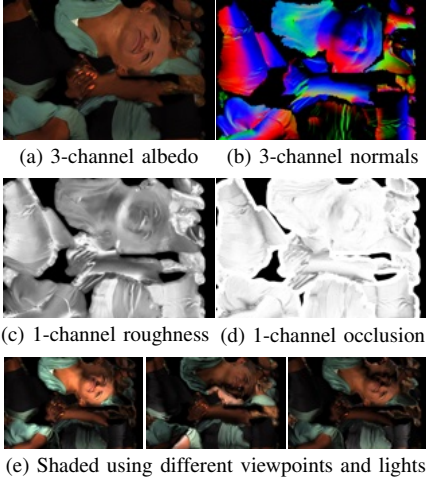


Fig. 13. Components of 8-channel texture map and images rendered to measure shaded distortion.

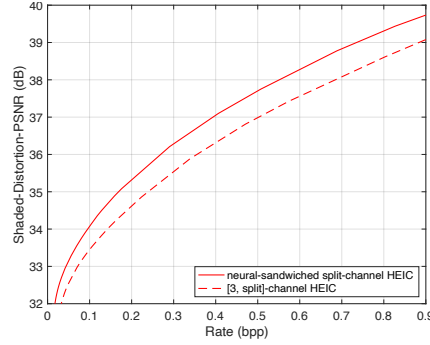


Fig. 14. R-D performance of compressing 8-channel texture map images with standard codec alone and neural-sandwiched.

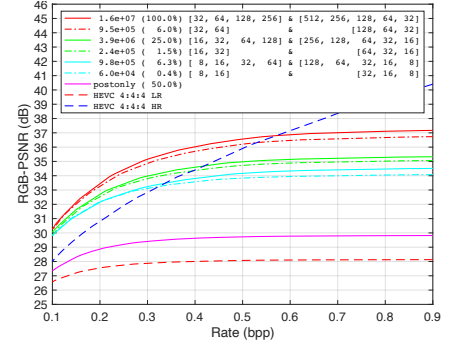


Fig. 15. R-D performance of compressing HR RGB images with LR codecs, with neural sandwiches of different complexity. Legend shows parameter counts (for pre-processor plus post-processor) for different U-Net configurations.

images with LR codecs, as detailed in subsection IV-B. While our default U-Net([32, 64, 128, 256]; [512, 256, 128, 64, 32]) has almost 8M parameters, many other U-Nets have many fewer parameters. Fig. 15 shows R-D results for the HR-LR application, for neural sandwiches with different complexities, assuming the pre- and post-processors have equal complexity. We have made trade-offs by reducing the number of blocks and/or the number of channels in each block. For example, U-Net([32, 64]; [128, 64, 32]), which has two encoder blocks and three decoder blocks, with their default number of channels, has only 491K parameters, a mere 6% of the parameters of the default U-Net, with less than 0.5 dB loss in R-D performance. In contrast, reducing complexity by reducing the number of channels has a less desirable trade-off.

Our experiments here show that massive reductions in the number of parameters are possible with little loss in performance. Of course, optimizing the hyper-parameters, such as the number of layers or the convolutional filter size, considering asymmetric models, using depth-separable convolutions, etc., can significantly improve complexity further (see, e.g., [44]). In the next section, on video experiments, we show that a U-Net([32]; [32, 32]), which we call our *slim* network with about 57K parameters, offers orders of magnitude reduction in complexity with little loss in R-D performance.

(Table S1 illustrates the parameter details of the UNet family explored in this paper.) Alternative architectures, which are more efficient in terms of MACs per pixel, are explored in [44].

VI. VIDEO COMPRESSION EXPERIMENTS

A. Codec Setup and Dataset

We generated a video dataset that consists of 10-frame clips of YUV video sequences from the AV2 Common Test Conditions [45] and their associated motion flows, calculated using UFlow [34]. We use a batch size of 8, i.e., 8 video clips in each batch. Each clip is processed during the dataset generation step such that it has 10 frames of size 256x256, selected from video of fps 20-40. HEVC is implemented using x265 (IPP., single reference frame, rdoq and loop filter on.)

within ffmpeg⁵. For each considered scenario, the model is trained for 1000 epochs, with a learning rate of $1e^{-4}$, and tested on 120 test video clips. RD plots are generated similar to Section IV. We report results in terms of YUV PSNR when using the ℓ_2 norm and “LPIPS (RGB) PSNR” when using LPIPS (refer to subsection VI-E for LPIPS particulars.)

B. Compressing 3-channel RGB Video with 1-channel (Gray-scale) Codecs

We start by considering the rate-distortion performance of the sandwich system on the toy example of transporting full-color video over a standard codec that can only carry gray-scale video (HEVC 4:0:0.) As we have already seen in Fig. 1, the sandwich-introduced modulation patterns are quite pronounced in terms of spatial extent and in terms of spatial frequency for the image compression version of this scenario. The toy example here is hence especially useful for examining (i) the temporal coherence of the sandwich-introduced patterns, (ii) the role of training with and without motion flows, and (iii) understanding the role of the network receptive field on the final compression scenario.

Video codecs provide the majority of compression gains by exploiting temporal redundancies via motion correspondences. Since the sandwich system uses modulation patterns for message passing and is deployed in a frame-independent fashion it is important that the patterns are introduced in a temporally consistent fashion that can be taken advantage of by standard codecs. This is clearly visible in Fig. 16, which shows three frames of compressed bottlenecks, sandwich-reconstructions, and originals. Observe that the patterns smoothly move with the scene objects that they are attached to. Through many such visuals we have noted that the networks operate in a translation robust manner with patterns moving with the objects and transitioning over motion/object discontinuities.

Fig. 17 demonstrates the significant improvements (+8.5 dB) that the neural-sandwiched HEVC 4:0:0 obtains over HEVC

⁵Observe that the rate for each clip reflects (i) an I-frame amortized over 10 frames and (ii) container format metadata though we tried to minimize the latter. The reader used to video rates where these two factors are amortized over hundreds of frames should expect higher reported rate numbers per-pixel.

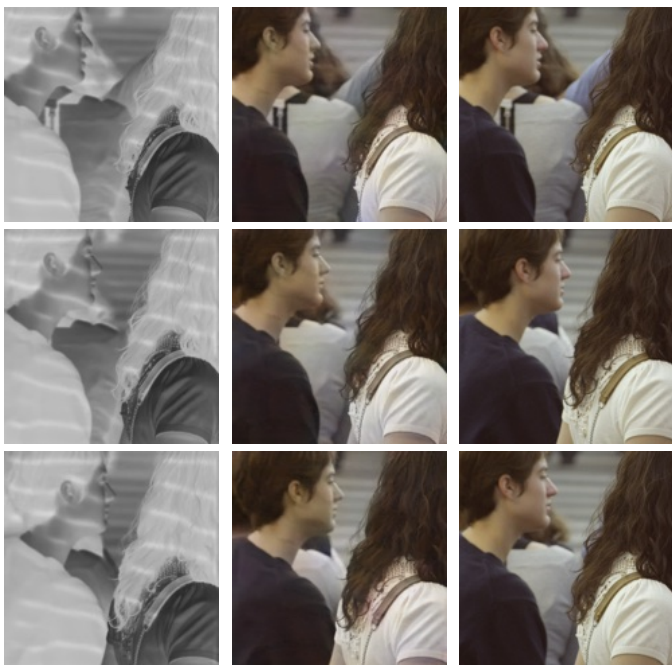


Fig. 16. Toy example with gray-scale codec (HEVC 4:0:0). The sandwich is used to transport full color video over a gray-scale codec. First, fifth, and tenth frames of compressed bottlenecks, final reconstructions by the post-processor, and original source videos are shown. Rate=0.10 bpp, PSNR=38.0 dB. The sandwich establishes temporally coherent modulation-like patterns on the bottlenecks through which the pre-processor encodes color that are then demodulated by the post-processor for a full-color result.

4:0:0. As an ablation study we trained a sandwich system using only the image codec proxy for the same scenario, shown in the Figure as “neural-sandwiched HEVC 4:0:0 (image-proxy)”. This system is unaware of the motion compensation process used by the codec and assumes the codec compresses video as a sequence of INTRA frames. Given a large enough training set, one expects this image-codec-proxy-trained sandwich to achieve translation robustness, as such a dataset depicts similar objects at many different translations. While performing worse than the video-codec-proxy-trained sandwich note that this version is also significantly better than HEVC 4:0:0 (~ 8 dB). This confirms the observation that well-trained pre/post-processors accomplish translation robustness, which is of fundamental importance in the video scenario.

Fig. 17 also shows the performance of the “slim” simplification, again, significantly outperforming HEVC 4:0:0. Note however that this model performs 3 dB below the full model. Since the slim model is restricted to the finest two resolutions as opposed to the full model’s four, its receptive field is significantly smaller than that of the full model. This in turn restricts its capacity to deploy spatially large patterns which appear to be advantageous in this problem.

C. Compressing 3-channel RGB Video with 3-channel Codecs

The rate-distortion performance of HEVC 4:4:4 and neural-sandwiched HEVC 4:4:4 are also included in Fig. 17. Over a broad rate-range the sandwich readily obtains $\sim 5\%$ improvements in rate at the same distortion. We have observed that the loop-filter proxy included as part of the video-codec proxy is performing better than the HEVC loop-filter. In effect, rather than compensating for the less-potent HEVC loop filter, the

neural-post-processor is trained assuming that the standard codec has a better loop-filter than it actually does. This leads to the neural-post-processor leaving some potential post-processing improvements on the table. Adjusting the loop-filter proxy to more closely mimic the HEVC loop filter is expected to marginally improve neural-sandwiched HEVC 4:4:4 results.

D. Compressing High Resolution (HR) RGB Video with Lower Resolution (LR) Codecs

In subsection IV-B we have seen that the sandwich can transport high-resolution (HR) images using lower-resolution (LR) codecs and obtain massive improvements. Using both JPEG and HEIC, the sandwich is significantly better over linear-down-codec-linear-up and linear-down-codec-neural-up transport schemes.

Given the results of subsection VI-B, *i.e.*, that the sandwich establishes temporally coherent message passing and continues to obtain massive improvements over the gray-scale codec in the video setting, we expect the sandwich to likewise extend subsection IV-B results to video. Not surprisingly we see this to be the case in Fig. 18, which shows the rate-distortion performance of transporting high-resolution (HR) video using a lower-resolution (LR) codec. Compared to HEVC 4:4:4 LR (Bicubic-down-HEVC4:4:4-Lanczos3-up) the sandwich obtains more than 6 dB improvements in YUV PSNR. The slim model is close with ~ 5 dB improvements. Note the parallels to the image case shown in Fig. 15 albeit in RGB PSNR.

Fig. 20 compares the visual quality of the reconstructed sandwich clips to that of HEVC 4:4:4 LR. The INTER coded fifth frame of each clip is shown. In the first-row note the significant amount of detail transported by the sandwich especially as depicted over the crowd and the building. HEVC 4:4:4 LR contains significant blur in those areas. The sandwich clip is 5 dB better at the same rate. In the second row note again not only the sharpness but the extra detail that the sandwich output contains especially toward the far-out points of the wire-structure. This detail, injected by the neural pre-processor and later demodulated by the neural post-processor, is simply missing from HEVC 4:4:4 LR. The sandwich is better by nearly 8 dB at the same rate.

E. Compressing RGB Video with an Alternative Perceptual Metric (LPIPS)

PSNR is well-known not to be a reliable metric for human-perceived visual quality. Alternatives such as the Learned Perceptual Image Patch Similarity (LPIPS) [46]–[48] and VMAF [49] are better suited for this task. While VMAF is not differentiable LPIPS is, and is thus readily suited to end-to-end gradient-based back-propagation. In this section we concentrate on LPIPS but also provide results on LPIPS-optimized sandwich networks on VMAF.

Since LPIPS is intended for RGB, the final decoded YUV video is first converted into RGB and then LPIPS is computed. In order to report results on an approximately similar scale to mean-squared-error results, we derived a fixed linear scaler for LPIPS so that for image vectors x, y ,

$$s\text{LPIPS}(x, y) \sim \|x - y\|^2, \text{ if } \|x - y\|^2 < \tau \quad (2)$$

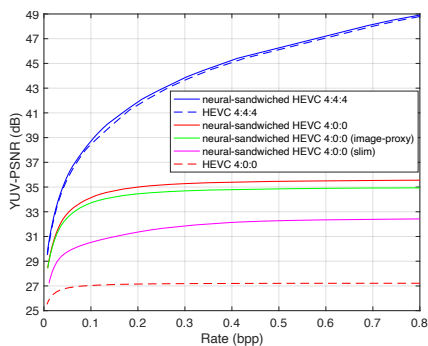


Fig. 17. Video rate-distortion performance of the YUV 4:0:0 sandwich and YUV 4:4:4 sandwich.

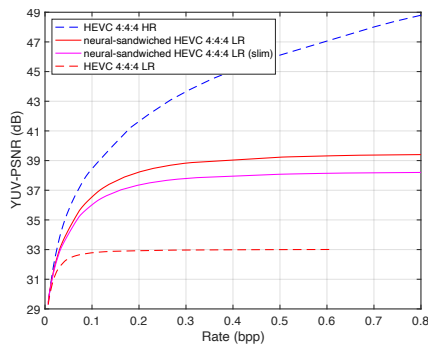


Fig. 18. Video rate-distortion performance of the YUV 4:4:4 low-resolution (LR) sandwich.

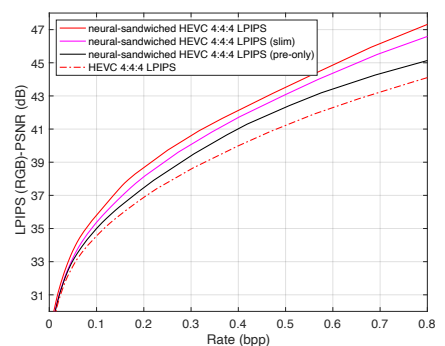


Fig. 19. Video rate-distortion performance of sandwich and HEVC with LPIPS.

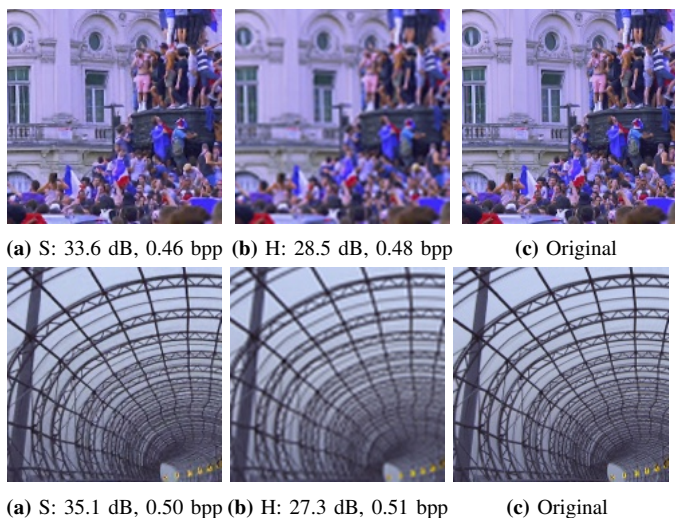


Fig. 20. Sandwich results for high-resolution video transport using a lower-resolution codec (HEVC 4:4:4 LR.) (a) Sandwich, (b) HEVC. HEVC 4:4:4 LR is implemented as Bicubic downsampling, followed by HEVC 4:4:4 compression, followed by Lamczos3 upsampling.

where τ is a small threshold and s is the LPIPS linear scalar. s is calculated once and is fixed for all results.

As mentioned in [47] when comparing single images there is the potential for adversarial attacks on LPIPS, *i.e.*, an image \tilde{x} , unrelated to x to a certain extent, can be designed to obtain $\text{LPIPS}(\tilde{x}, y) \sim \text{LPIPS}(x, y)$. One hence has to consider the possibility of an optimization scheme “hacking” the metric. To that end [47] recommends an ensemble LPIPS score where rather than calculating a single LPIPS score on the x, y pair, one calculates several scores by randomly applying slight geometric and intensity transformations to the pair. These scores are then averaged for the ensemble score. This ensemble score is shown to be robust to adversarial attacks.

By their very nature, video clips are typically versions of scenes with geometric deformations (scene motion) and color/brightness changes (scene lighting changes.) We hence report the LPIPS loss of the n^{th} clip as the average of the LPIPS losses over its $T = 10$ frames, *i.e.*,

$$D_n = \frac{1}{T} \sum_{t=0}^{T-1} s\text{LPIPS}(x_n(t), y_n(t)), \quad (3)$$

where $x_n(t), y_n(t)$ are the t^{th} frames of the decoded and

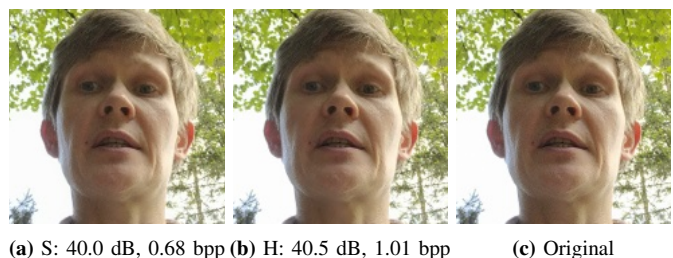
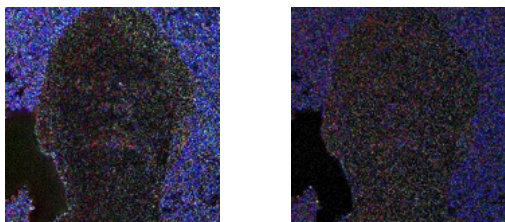


Fig. 21. Sandwich results for the LPIPS scenario (5^{th} frame in each clip is shown.). (a) Sandwiched HEVC results, (b) HEVC results. It is difficult to find significant differences among the results at 40 dB LPIPS-PSNR. The sandwich clip has $\sim 32\%$ lower rate than the HEVC clip.

original clips respectively. We expect this averaging process to help improve the robustness of the score but we also (i) evaluate visual quality, (ii) show that the slim network obtains similar performance (with substantially reduced parameters the slim network has less room for hacking the metric,) and (iii) report VMAF results of the LPIPS-optimized sandwich on an especially meaningful scenario. In what follows we report “LPIPS (RGB) PSNR” which is the PSNR of the relevant averaged LPIPS loss.

Fig. 19 compares the rate-distortion performance of the neural-sandwiched HEVC against HEVC with clip distortion measured via [3]. Observe that the sandwich with the full model obtains $\sim 30\%$ improvements in rate at the same LPIPS quality. The slim model closely tracks these results with $\sim 20\text{--}25\%$ improvements. Lastly we see that a pre-processor only variant, where we have disabled the neural-post-processor and trained only a pre-processor, obtains $\sim 10\text{--}15\%$. Considering that a new generation standard codec typically improves $\sim 30\%$ over the previous generation, one can see that *the full and slim networks are offering generational improvements assuming LPIPS accurately represents human-perceived quality*. As interestingly, the pre-processor-only result indicates that a video streaming service can potentially reduce bandwidth by 10-15% in a way transparent to its users’ decoders.

In order to vet the correspondence of LPIPS and visual quality we subjectively examined the clips of the neural-sandwiched HEVC and HEVC decoded at the same LPIPS quality. At high LPIPS quality levels we found no significant differences among sandwich, HEVC, and original clips. A sample from such a clip is shown in Fig. 21, where all three



(a) S: 40.0 dB, 0.68 bpp (b) H: 40.5 dB, 1.01 bpp

Fig. 22. Absolute errors (15x amplified) of the sandwich and HEVC frames depicted in Fig. 21. The sandwich frame has higher errors which are nevertheless difficult to perceive in Fig. 21 as the visually important structures such as edges are well-preserved.

samples look similar while the sandwich clip has $\sim 32\%$ less rate than the HEVC clip. Fig. 22 in fact shows that the sandwich output contains more absolute-errors. It is nevertheless hard to discern visual quality differences. At intermediate to low quality levels it is difficult to pick between the sandwich and HEVC clips though differences to the original clip become more noticeable (Fig. S7). Quality degrades in expected ways and the sandwich retains rate improvements at the same LPIPS quality until the low quality regime. Supplementary section VIII-G explores the types of clip and region statistics where LPIPS may be enabling the gains.

Last but not least, especially since VMAF is typically used to gauge quality in the streaming scenario, we also evaluated the VMAF scores of the sandwich pre-processor-only network. Fig. S6 shows that the pre-processor-only network (optimized for LPIPS) obtains $\sim 10\%$ improvements in rate at the same VMAF quality. While we have not done so this agreement between the metrics suggests training the sandwich for LPIPS, evaluating the models during training for VMAF as well, and picking a model that has acceptable improvements for both.

VII. DISCUSSION AND CONCLUSION

In this paper, we have proposed sandwiching standard image and video codecs between neural pre- and post-processors, trained through a proxy. Remarkably, the neural pre- and post-processor learn to communicate *source* images to each other by sending *coded* images through the standard codec. The coded images may have fewer channels, lower resolution, and lower dynamic range than the source images that they represent. Yet, even though the coded images are quantized by the standard codec to a PSNR commensurate with the bitrate, the source images achieve superior fidelity in the same distortion measure or in an alternative distortion measure, at that bitrate.

While the sandwich architecture improves upon the standard codec’s compression of typical color images under the MSE distortion, the strength of the architecture is that it allows the standard codec to adapt to coding non-typical images under possibly non-typical distortion measures. We have provided an extensive set of simulation results clearly demonstrating the value of the sandwich. Nevertheless our examples are not intended to be exhaustive or definitive. For instance, we did not explore adaptation to medical, hyper-spectral, or other multi-channel imagery. (However, see [18] for applications of sandwiching to multi-view compression.) Nor did we explore adaptation to higher temporal resolution. We note these as areas for future research.

Our differentiable image codec proxy is modeled after JPEG. Yet our experimental results consistently show that the pre- and post-processors trained with this proxy can be used to sandwich not only JPEG but also — without retraining — HEIC (and in fact HEVC with the an adaptation of this proxy to video) with significant rate-distortion gains compared to the non-sandwiched codec. Furthermore, the sandwich gains have been shown to extend to wrapping VVC and AV1 codecs with low-complexity post-processors (500 MACs) [44]. That is a level of complexity that can allow neural processing to be included in next generation video compression standards such as the upcoming AV2.

As standard codecs continue to advance it would be beneficial to design a differentiable codec proxy that is closer to the target standard codec for even better performance. Of course one day the standard codec itself may be differentiable. For example, it may become an end-to-end neural codec. It is important to note that even in that case, the sandwich architecture would remain a valid way to adapt, or *fine-tune*, the standard codec to alternative source types and distortion measures (Proposition 1.)

We advocate that future image and video codec standards be designed with sandwiching in mind. As we have seen, it is possible to adapt codecs to altogether new image and distortion types. This can for example be accomplished by specifying a simple neural post-processor in the compressed bit stream header. More generally, such a technique could be used at any level of the bit stream, *e.g.*, GOP-level, picture-level, or block-level, to signal that specific neural processors be used in the decoder, possibly matched to specific neural processors used in the encoder, which have been trained to communicate with each other through learned neural codes. In these ways, we advocate for making standard codecs more universal and thus more broadly applicable to alternative source types and distortion measures, such as in graphics, augmented/virtual reality, medical imaging, multi-modal sensing for autonomous driving, and so forth.

Our results clearly indicate that MSE-optimized codecs can be easily repurposed to other metrics and scenarios. Hence, despite the reputation of MSE as an inadequate visual quality metric, the call for replacing it with other metrics in standard codec design may not be clear-cut. Given its ease of optimization in incrementally furthering individual compression tools it may in fact remain the metric of choice in designing inner compression engines to be generalized as needed by sandwiching.

The code base for this work is open-sourced at https://github.com/google/sandwiched_compression [50].

REFERENCES

- [1] W. B. Pennebaker and J. L. Mitchell, *JPEG Still Image Data Compression Standard*. Van Nostrand Reinhold, 2022.
- [2] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, “Overview of the H.264/AVC video coding standard,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, 2003.
- [3] D. Mukherjee, J. Bankoski, A. Grange, J. Han, J. Koleszar, P. Wilkins, Y. Xu, and R. Bultje, “The Latest Open-Source Video Codec VP9 - an Overview and Preliminary Results,” in *Picture Coding Symp. (PCS)*. IEEE, 2013, pp. 390–393.
- [4] M. Wien, *High Efficiency Video Coding: Coding Tools and Specification*. Springer Publishing Company, Incorporated, 2014.

- [5] J. Han, B. Li, D. Mukherjee, C.-H. Chiang, A. Grange, C. Chen, H. Su, S. Parker, S. Deng, U. Joshi *et al.*, “A technical overview of AV1,” *Proceedings of the IEEE*, vol. 109, no. 9, pp. 1435–1462, 2021.
- [6] B. Bross, Y.-K. Wang, Y. Ye, S. Liu, J. Chen, G. J. Sullivan, and J.-R. Ohm, “Overview of the versatile video coding (VVC) standard and its applications,” *IEEE TCSVT*, vol. 31, no. 10, pp. 3736–3764, 2021.
- [7] 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.
- [8] J. Balle, P. A. Chou, D. Minnen, S. Singh, N. Johnston, E. Agustsson, S. J. Hwang, and G. Toderici, “Nonlinear Transform Coding,” *IEEE J. Selected Topics in Signal Processing*, pp. 1–1, 2020.
- [9] Z. Guo, Z. Zhang, R. Feng, and Z. Chen, “Causal Contextual Prediction for Learned Image Compression,” *IEEE TCSVT*, pp. 1–1, 2021.
- [10] G. Lu, W. Ouyang, D. Xu, X. Zhang, C. Cai, and Z. Gao, “DVC: An End-To-End Deep Video Compression Framework,” in *CVPR*, 2019.
- [11] Z. Hu, G. Lu, and D. Xu, “Fvc: A new framework towards deep video compression in feature space,” in *CVPR*, 2021, pp. 1502–1511.
- [12] O. Rippel, A. G. Anderson, K. Tatwawadi, S. Nair, C. Lytle, and L. Bourdev, “ELF-VC: Efficient Learned Flexible-Rate Video Coding,” in *CVPR*, 2021, pp. 14479–14488.
- [13] D. Minnen and N. Johnston, “Advancing the rate-distortion-computation frontier for neural image compression,” in *ICIP*, 2023, pp. 2940–2944.
- [14] F. Mentzer, G. D. Toderici, M. Tschannen, and E. Agustsson, “High-Fidelity Generative Image Compression,” in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 11913–11924.
- [15] O. G. Guleryuz, P. A. Chou, H. Hoppe, D. Tang, R. Du, P. Davidson, and S. Fanello, “Sandwiched Image Compression: Wrapping Neural Networks Around a Standard Codec,” in *ICIP*, 2021.
- [16] —, “Sandwiched image compression: Increasing the resolution and dynamic range of standard codecs,” in *2022 Picture Coding Symposium (PCS)*, 2022, pp. 175–179.
- [17] B. Isik, O. Guleryuz, D. Tang, J. Taylor, and P. Chou, “Sandwiched video compression: Efficiently extending the reach of standard codecs with neural wrappers,” in *ICIP*. IEEE, 2023, pp. 3757–3761.
- [18] Y. Hu, O. G. Guleryuz, P. A. Chou, D. Tang, J. Taylor, R. Maxham, and Y. Wang, “One-click upgrade from 2D to 3D: Sandwiched RGB-D video compression for stereoscopic teleconferencing,” in *submission*.
- [19] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, “Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising,” *IEEE Trans. Image Processing*, vol. 26, no. 7, pp. 3142–3155, 2017.
- [20] C. Tian, L. Fei, W. Zheng, Y. Xu, W. Zuo, and C.-W. Lin, “Deep Learning on Image Denoising: an Overview,” *Neural Networks*, vol. 131, pp. 251 – 275, 2020.
- [21] H. Vu, G. Cheung, and Y. C. Eldar, “Unrolling of Deep Graph Total Variation for Image Denoising,” in *IEEE ICASSP*, 2021, pp. 2050–2054.
- [22] P. Svoboda, M. Hradis, D. Barina, and P. Zemečik, “Compression Artifacts Removal Using Convolutional Neural Networks,” *ArXiv*, vol. abs/1605.00366, 2016.
- [23] T. Kim, H. Lee, H. Son, and S. Lee, “SF-CNN: A Fast Compression Artifacts Removal Via Spatial-to-Frequency Convolutional Neural Networks,” in *IEEE ICIP*, 2019, pp. 3606–3610.
- [24] J. Niu, “End-to-End JPEG Decoding and Artifacts Suppression Using Heterogeneous Residual Convolutional Neural Network,” *Int’l Joint Conf. Neural Networks (IJCNN)*, pp. 1–8, 2020.
- [25] Y. Li, D. Liu, H. Li, L. Li, Z. Li, and F. Wu, “Learning a Convolutional Neural Network for Image Compact-Resolution,” *IEEE Trans. Image Processing*, vol. 28, no. 3, pp. 1092–1107, 2019.
- [26] P. Eusébio, J. Ascenso, and F. Pereira, “Optimizing an Image Coding Framework With Deep Learning-Based Pre- and Post-Processing,” in *European Signal Processing Conf. (EUSIPCO)*, 2021, pp. 506–510.
- [27] K. Qiu, L. Yu, and D. Li, “Codec-Simulation Network for Joint Optimization of Video Coding With Pre- and Post-Processing,” *IEEE Open J. Circuits and Systems*, vol. 2, pp. 648–659, 2021.
- [28] Y. Andreopoulos, “Neural Pre and Post-Processing for Video Encoding With AVC, VP9, and AV1,” in *AOM Research Symp*, 2022.
- [29] 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 *IEEE/CVF CVPR Workshops*, 2020, pp. 565–569.
- [30] C. Segall and A. Katsaggelos, “Pre- and Post-Processing Algorithms for Compressed Video Enhancement,” in *Asilomar Conf. Signals, Systems and Computers*, vol. 2, 2000, pp. 1369–1373 vol.2.
- [31] S. Park and R. Gray, “Sigma-delta modulation with leaky integration and constant input,” *IEEE Transactions on Information Theory*, vol. 38, no. 5, pp. 1512–1533, 1992.
- [32] O. Guleryuz and M. Orchard, “On the dpcm compression of gaussian autoregressive sequences,” *IEEE Transactions on Information Theory*, vol. 47, no. 3, pp. 945–956, 2001.
- [33] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation,” in *Medical Image Computing and Computer-Assisted Intervention*, 2015, pp. 234–241.
- [34] S. K. Lodha, A. Pang, R. E. Sheehan, and C. M. Wittenbrink, “Uflow: Visualizing uncertainty in fluid flow,” in *Proceedings of Seventh Annual IEEE Visualization’96*. IEEE, 1996, pp. 249–254.
- [35] “The Oxford-IIIT Pet Dataset,” https://www.tensorflow.org/datasets/catalog/oxford_iiit_pet, 2022.
- [36] “Dataset for the Challenge on Learned Image Compression 2020,” <http://www.tensorflow.org/datasets/catalog/clci>, 2022.
- [37] S. W. Hasinoff, D. Sharlet, R. Geiss, A. Adams, J. Barron, F. Kainz, J. Chen, and M. Levoy, “Burst Photography for High Dynamic Range and Low-Light Imaging on Mobile Cameras,” *ACM TOG*, 2016.
- [38] K. Guo *et al.*, “The Relightables: Volumetric Performance Capture of Humans With Realistic Relighting,” in *ACM TOG*, 2019.
- [39] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 2022.
- [40] E. Agustsson and R. Timofte, “NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study,” in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.
- [41] Y.-L. Liu, W.-S. Lai, Y.-S. Chen, Y.-L. Kao, M.-H. Yang, Y.-Y. Chuang, and J.-B. Huang, “Single-Image HDR Reconstruction by Learning to Reverse the Camera Pipeline,” in *IEEE CVPR*, 2020.
- [42] Wikipedia contributors, “Normal mapping - wikipedia.”
- [43] A. C. Beers, M. Agrawala, and N. Chaddha, “Rendering from compressed textures,” in *SIGGRAPH*, 1996, p. 373–378.
- [44] Y. Hu, C. Zhang, O. G. Guleryuz, D. Mukherjee, and Y. Wang, “Standard compatible efficient video coding with jointly optimized neural wrappers,” in *submission to Data Compression Conference*. IEEE, 2024.
- [45] “Aom common test conditions v2.0,” http://aomedia.org/docs/CWG-B075c_AV2_CTC_v2.pdf, accessed: 2022-07-11.
- [46] R. Zhang *et al.*, “The unreasonable effectiveness of deep features as a perceptual metric,” in *IEEE CVPR*, 2018, pp. 586–595.
- [47] M. Kettunen, E. Härkönen, and J. Lehtinen, “E-LPIPS: robust perceptual image similarity via random transformation ensembles,” *CoRR*, vol. abs/1906.03973, 2019.
- [48] K. Ding, K. Ma, S. Wang, and E. P. Simoncelli, “Comparison of full-reference image quality models for optimization of image processing systems,” *IJCV*, vol. 129, no. 4, p. 1258–1281, apr 2021.
- [49] Z. Li *et al.*, “Toward a practical perceptual video quality metric,” *The Netflix Tech Blog*, vol. 6, no. 2, p. 2, 2016.
- [50] “Sandwiched Compression Software,” https://github.com/google/sandwiched_compression, 2024.
- [51] X. Luo, H. Talebi, F. Yang, M. Elad, and P. Milanfar, “The Rate-Distortion-Accuracy Tradeoff: JPEG Case Study,” 2020.
- [52] 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 *ICLR*, 2016.
- [53] “TensorFlow API: tf.stopgradient,” https://www.tensorflow.org/api_docs/python/tf/stop_gradient, 2022.
- [54] Z. He and S. Mitra, “A Unified Rate-Distortion Analysis Framework for Transform Coding,” *IEEE TCSVT*, vol. 11, pp. 1221–1236, 2001.
- [55] A. Said, M. K. Singh, and R. Poureza, “Differentiable bit-rate estimation for neural-based video codec enhancement,” in *2022 Picture Coding Symposium (PCS)*, 2022, pp. 379–383.
- [56] T. Cover and J. Thomas, *Elements of Information Theory*. Wiley, 2012.
- [57] Wikipedia contributors, “Universal approximation theorem,” 2023.
- [58] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang, “The expressive power of neural networks: A view from the width,” in *NeurIPS*, vol. 30, 2017.
- [59] S. Park, C. Yun, J. Lee, and J. Shin, “Minimum width for universal approximation,” in *ICLR*, 2021.
- [60] P. Chou, T. Lookabaugh, and R. Gray, “Entropy-constrained vector quantization,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 1, pp. 31–42, 1989.

VIII. SUPPLEMENTARY MATERIAL

A. Quantizer and Rate Proxies (Suppl. to Sec. III-A)

Various differentiable *quantizer proxies* are possible (Fig. S1). Luo *et al.* [51] 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., [52]–[7]) use either additive uniform noise $Q(X_i) = X_i + U_i\Delta$, where U_i is i.i.d. $\sim \text{unif}(-1/2, 1/2)$, or a “straight-through” quantizer $Q(X_i) = X_i + U_i\Delta$, where $U_i\Delta = \text{stop_gradient}(\text{round}(X_i/\Delta) - X_i/\Delta)\Delta$ is the true quantization noise and $\text{stop_gradient}(\cdot)$ is the identity mapping but stops the gradient of its output from being back-propagated to its argument [53]. In all cases, the derivative of $\hat{X}_i = Q(X_i)$ with respect to X_i is nonzero almost everywhere. This allows non-trivial gradients of the end-to-end distortion $d(S, \hat{S})$ with respect to the parameters of the networks using the chain rule and back-propagation. These formulations also allow the stepsize Δ to receive gradients, which is necessary to properly minimize the Lagrangian. 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 [54] 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

$$R(B) = \sum_k R(X^{(k)}) = a \sum_{k,i} \log\left(1 + \left|x_i^{(k)}\right|/\Delta\right) + b. \quad (4)$$

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.*:

$$a = \frac{R_{\text{JPEG}}(B, \Delta)}{\sum_{k,i} \log\left(1 + \left|x_i^{(k)}\right|/\Delta\right)}. \quad (5)$$

This ensures that the differentiable function $R(B)$ is exactly equal to $R_{\text{JPEG}}(B, \Delta)$ 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 pre-processor, and with respect to the stepsize Δ , can be computed using back-propagation. An alternative rate proxy is given in [55].

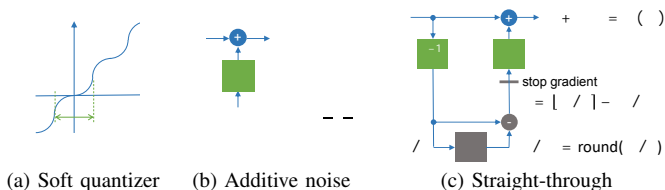


Fig. S1. Possible quantizer proxies.

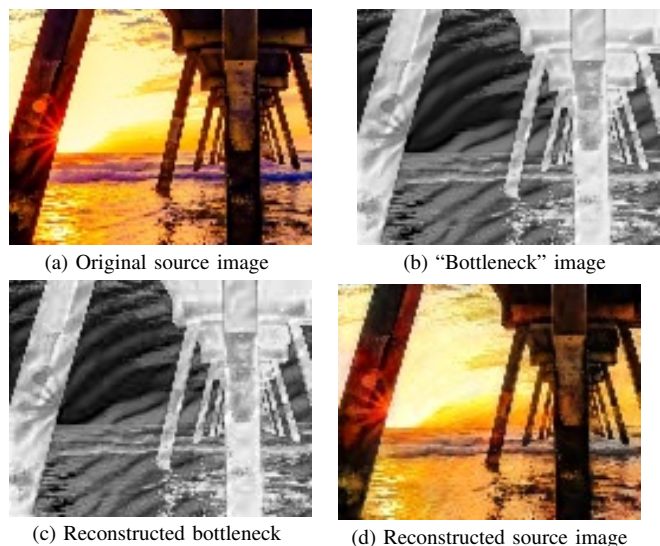


Fig. S2. Even with a simple codec (here JPEG 4:0:0, a single-channel grayscale codec), the sandwich architecture can accomplish unconventional results. Whereas 4:0:0 usually accommodates only luminance, the neural pre-processor is able to encode a full RGB image into a single-channel luminance-like image of neural codes. The neural codes are low-frequency dither-like patterns that modulate the color information yet also survive JPEG compression. At the decoding end, the neural post-processor demodulates the patterns to recover the original color while also achieving deblocking.

B. HR and HDR Adaptations (Suppl. to Sec. III-A)

In the HR problem, the RGB $H \times W \times 3$ source images have source bit depth $d = 8$. Thus they have the standard dynamic range, $[0, 255]$. However, the bottleneck images have lower spatial resolution, $H/2 \times W/2 \times 3$. In our work, the resampler in the pre-processor comprises bicubic filtering and 2x downsampling; the resampler in the post-processor comprises Lanczos3 interpolation of the half-resolution images back to full-resolution. Similar down- and up-sampling is done in [26].

In the HDR problem, the source images have dynamic range $[0, 2^d - 1]$, where d is the source bit depth. The bottleneck images have dimensions that match the source images: $H \times W \times 3$ but are restricted to the standard dynamic range $[0, 255]$. Since the codec proxy does not pass any information outside of this range, the pre-processor produces images in this range.

In both the HR and HDR problems, the sandwiched codecs operate in 4:4:4 mode without a color transform. Regardless, the baseline (non-sandwiched) codecs that we compare to use the $\text{RGB} \leftrightarrow \text{YUV}$ transform when it is beneficial for them in an R-D sense: In the HR scenario they use the color transform; in HDR they encode RGB directly.

C. More RGB-Grayscale Image Results (Suppl. to Sec. IV-A)

Fig. S2 provides another example of using a sandwich to transport 3-channel RGB images through a 1-channel (grayscale) codec.

D. More HR-LR Image Results (Suppl. to Sec. IV-B)

Fig. S3 and Fig. S4 provide further examples of using a sandwich to transport high resolution (HR) images through a lower resolution (LR) standard codec.

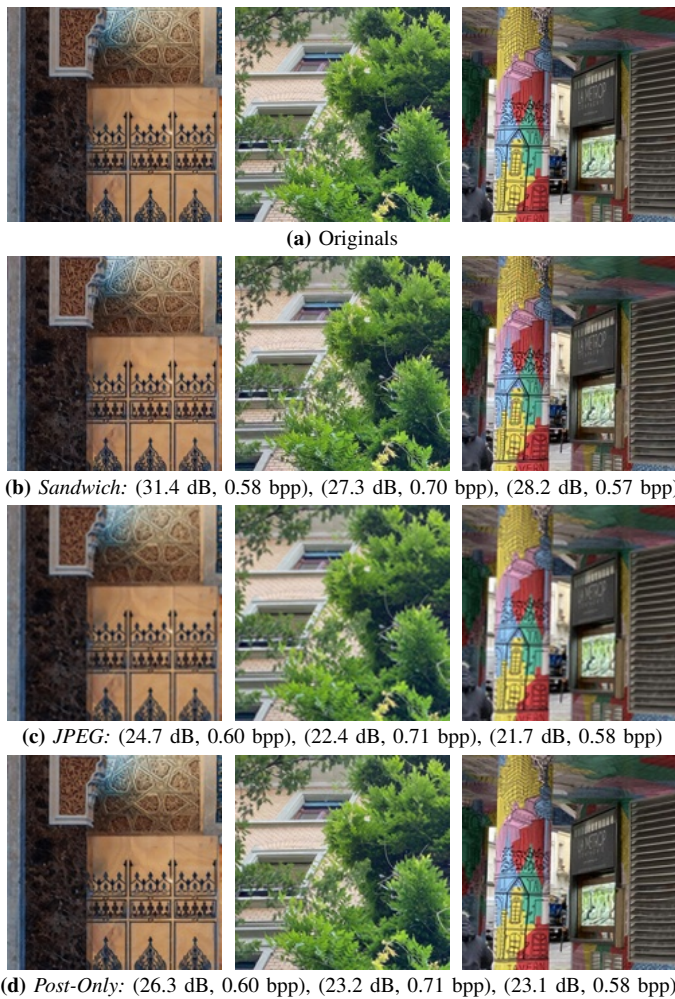


Fig. S3. Super-resolution sandwich: Original 256×256 source images and reconstructions by sandwich, JPEG with linear upsampling, and JPEG enhanced with neural post-processing respectively. With the sandwich visually relevant ornaments/textures are preserved, images are sharper in a way that matches the originals, and text in the scene is easier to read. Beyond significantly improved visual quality the sandwich obtains substantial dB improvements (+5.1 dB, +4.1 dB, +5.1 dB over neural post-processing) at the same rate.

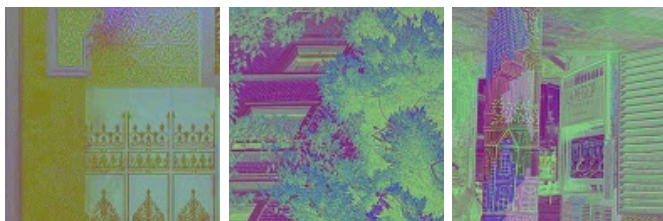


Fig. S4. 128×128 reconstructed bottleneck images for the super-resolution sandwich results in Fig. S3 [enlarged for clarity]. Observe that while the bottlenecks appear aliased, noisy etc., the sandwich post-processor has correctly demodulated this noise in the final pictures.

E. More Normal Map Image Results (Suppl. to Sec. IV-D)

Fig. S5 shows results of compressing normal maps with sandwiched JPEG. Compare Fig. 12, which shows corresponding results for sandwiched HEIC. The gains due to sandwiching are preserved in either case.

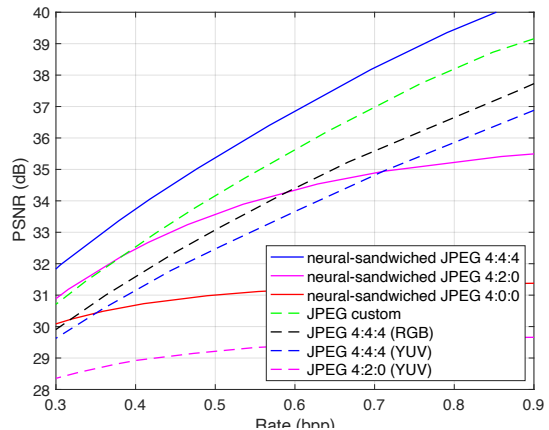


Fig. S5. R-D performances of compressing normal map images with JPEG and neural-sandwiched JPEG, in various formats.

U-Net hyperparameters		number of parameters	MACs per pixel
encoder	decoder		
[32, 64, 128, 256]	[512, 256, 128, 64, 32]	7847491	213943
[32, 64]	[128, 64, 32]	472387	112531
[16, 32, 64, 128]	[256, 128, 64, 32, 16]	1963043	53981
[16, 32]	[64, 32, 16]	118691	28619
[8, 16, 32, 64]	[128, 64, 32, 16, 8]	491347	13743
[8, 16]	[32, 16, 8]	29971	7399
[32]	[32, 32]	57219	43347

TABLE S1
U-NET COMPLEXITY FOR VARIOUS HYPERPARAMETERS.
FIXED HYPERPARAMETERS INCLUDE $C_{in} = C_{out} = 3$,
FILTERS SIZE = 3×3 , AND LAYERS PER BLOCK = 2.

F. Further Information on Complexity (Suppl. to Sec. V)

Table S1 illustrates the parameter details of the UNet family explored in this paper.

G. More LPIPS Video Results (Suppl. to Sec. VI-E)

Fig. S7 provides further examples of using a sandwich optimized for LPIPS to transport images through a standard codec. Whereas Fig. 21 showed 32% rate savings at a quality visually close to the original, Fig. S7 shows corresponding rate saving at lower visual quality levels.

To better understand where the sandwich with LPIPS is getting improvements we considered clips where rate gains (at the same LPIPS quality) were high, intermediate, and low. Fig. S8 shows sample clips from each group. As illustrated the gains are strongly tied to the high frequency and texture content of the scene. While scenes dense with such content (top row) have the largest improvements, scenes with even moderate amounts of high frequency structures (middle row) induce noticeable gains.

Fig. S6 shows that the pre-processor-only network (optimized for LPIPS) obtains $\sim 10\%$ improvements in rate at the same VMAF quality.

H. Theoretical Limits of Neural Sandwiching (Suppl. to Sec. III)

In this section, we explore the theoretical limits of neural sandwiching. In particular, we prove the following Proposition,

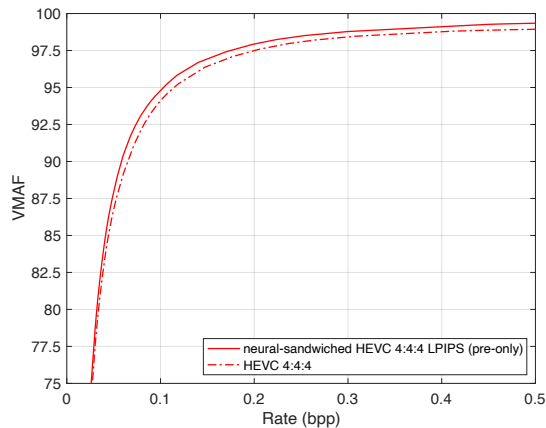


Fig. S6. VMAF scores of the sandwich trained using LPIPS (pre-processor only) and HEVC. Over a broad range sandwich shows $\sim 10\%$ improvements in rate at same VMAF quality.

which is stronger than Prop. 1 in Section II in that it includes an arbitrary permutation.

Proposition 2. Let X be a \mathbb{R}^n -valued bounded source, let d be a distortion measure, and let $D(R)$ be the operational distortion-rate function for X under d . For any $\epsilon > 0$, let $(\alpha^*, \beta^*, \gamma^*)$ be a rate- R codec for X achieving $D(R)$ within $\epsilon/2$. Let (α, β, γ) be a regular codec (e.g., a standard codec, possibly designed for a different source and different distortion measure) with bounded codelengths. Then for any permutation π , there exist neural pre- and post-processors f and g such that the codec sandwich $(\alpha \circ f, g \circ \beta, \gamma)$ has expected distortion at most $D(R) + \epsilon$ and expected rate at most $R + D(p||q) + \epsilon$, where $p(k) = P(\{\alpha^*(X) = k\})$ and $q(k) = 2^{-|\gamma(\pi(k))|}$.

Remark. $D(p||q)$ is the worst-case rate penalty for having to re-use the entropy coder from the standard codec.

Remark. By optimizing over the permutation π , the rate penalty $D(p||q)$ may be minimized. Indeed, the penalty is minimized when the codelengths $|\gamma(\pi(k))|$ are sorted in the same order as the codelengths $|\gamma^*(k)|$. (See Prop. 3)

First some definitions: A \mathbb{R}^n -valued source X is a random vector (e.g., an image), where n is the dimension of the source (e.g., the number of pixels in the image). The source is bounded if for some finite bound b , $X \in [-b, b]^n$ with probability 1. A codec for X is given by a triple (α, β, γ) , where the encoder $\alpha : \mathbb{R}^n \rightarrow \mathcal{K}$ maps each source vector $x \in \mathbb{R}^n$ to a index $k \in \mathcal{K}$, the decoder $\beta : \mathcal{K} \rightarrow \mathbb{R}^n$ maps each index $k \in \mathcal{K}$ to a reproduction vector $\hat{x} \in \mathbb{R}^n$; and the lossless encoder $\gamma : \mathcal{K} \rightarrow \mathcal{C}$ invertibly maps each element of \mathcal{K} to a binary string in a codebook $\mathcal{C} \subset \{0, 1\}^*$ of variable-length binary strings satisfying the Kraft inequality, $\sum_{k \in \mathcal{K}} 2^{-|\gamma(k)|} \leq 1$, where $|s|$ denotes the length in bits of the binary string s . The Kraft inequality guarantees the existence of a prefix-free, and hence uniquely decodable (i.e., invertible), binary lossless encoder γ [56]. Alternatively γ may be considered an arithmetic coder or other entropy coder with nominal codelengths $\{|\gamma(k)|\}$. A quantization cell is the set

$\{x : \alpha(x) = k\}$ of source vectors encoding to index k . A codec is regular if each of its quantization cells is a non-degenerate polytope (i.e., the intersection of half-spaces with non-empty interior). Codecs based on scalar quantization (e.g., transform coders) as well as nearest-neighbor quantizers are all regular. A distortion measure $d : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^+$ maps a source vector x and its reproduction, say $\hat{x} = \beta(\alpha(x))$, to a non-negative number. The expected distortion of the codec is

$$D(\alpha, \beta) = E[d(X, \beta(\alpha(X)))], \quad (6)$$

and the expected rate of the codec is

$$R(\alpha, \gamma) = E[|\gamma(\alpha(X))|]. \quad (7)$$

The operational distortion-rate function for X under d is

$$D(R) = \inf_{\alpha, \beta, \gamma} \{D(\alpha, \beta) : R(\alpha, \gamma) \leq R\}. \quad (8)$$

Proof. Given the \mathbb{R}^n -valued bounded source X , the distortion measure d , the operational distortion-rate function $D(R)$ for X under d , and any $\epsilon > 0$, let $(\alpha^*, \beta^*, \gamma^*)$ be a near-optimal codec at rate R , such that

$$D(\alpha^*, \beta^*) \leq D(R) + \epsilon/2 \quad (9)$$

$$R(\alpha^*, \gamma^*) \leq R. \quad (10)$$

Now given a regular codec (α, β, γ) generally not for the source X but for some other source Y , which may be \mathbb{R}^m -valued, we need to find a neural pre-processor $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ and a neural post-processor $g : \mathbb{R}^m \rightarrow \mathbb{R}^n$ such that the composition $\alpha \circ f$ and the composition $g \circ \beta$ satisfy

$$D(\alpha \circ f, g \circ \beta) \leq D(R) + \epsilon \quad (11)$$

$$R(\alpha \circ f, \gamma) \leq R + D(p||q) + \epsilon, \quad (12)$$

where $p(k) = P(\{\alpha(f(X)) = k\})$ and $q(k) = 2^{-|\gamma(\pi(k))|}$.

To accomplish this, we will find f and g so that $\alpha \circ f$ approximates the near-optimal encoder α^* , and $g \circ \beta$ approximates the near-optimal decoder β^* . Optionally we may also find a permutation $\pi : \mathcal{K} \rightarrow \mathcal{K}$ such that the composition $\gamma \circ \pi$ approximates the optimal lossless encoder γ^* . Such a permutation minimizes the bound $D(p||q)$.

First we will prove (11) by showing that if the approximation is good enough, then the expected distortion increases by at most $\epsilon/2$, namely

$$D(\alpha \circ f, g \circ \beta) \leq D(\alpha^*, \beta^*) + \epsilon/2. \quad (13)$$

Together, (9) and (13) imply (11).

To show (13), first we define an “ideal” pre-processor f^* such that for all $x \in \mathbb{R}^n$, $f^*(x) = y_k$ whenever $\alpha^*(x) = k$, where y_k is a point in the interior of the quantization cell $\{y : \alpha(y) = \pi(k)\}$. (The cell has an interior because (α, β, γ) is assumed to be regular.) We also define an “ideal” post-processor g^* such that for all $k \in \mathcal{K}$: $g^*(\beta(\pi(k))) = \beta^*(k)$. The definition of $g^*(\hat{y})$ for values of \hat{y} not in the discrete set $\{\beta(\pi(k)) : k \in \mathcal{K}\}$ are arbitrary, as β produces values only in this set. With these definitions, it can be seen that $\alpha(f^*(x)) = \pi(\alpha^*(x))$ and $g^*(\beta(\pi(\alpha^*(x)))) = \beta^*(\alpha^*(x))$ and hence

$$g^*(\beta(\alpha(f^*(x)))) = \beta^*(\alpha^*(x)), \quad (14)$$

i.e., $\pi^{-1} \circ \alpha \circ f^*$ emulates α^* and $g^* \circ \beta \circ \pi$ emulates β^* . Thus $D(\alpha \circ f^*, g^* \circ \beta) = D(\alpha^*, \beta^*)$.

We now argue that there exists a neural pre-processor f sufficiently close to f^* . Indeed, by a Universal Approximation Theorem for neural networks [57]–[59], there is a neural network f arbitrarily close to f^* in L_2 , i.e., for all $\delta > 0$ there exists f such that $E[\|f(X) - f^*(X)\|^2] < \delta$. A fortiori, as convergence in L_2 implies convergence in probability, for all $\delta > 0$, there exists f and a set Ω with $P(\Omega) > 1 - \delta$ such that for all $x \in \Omega$, $\|f(x) - f^*(x)\| < \delta$. Thus whenever $x \in \Omega$ and $\alpha^*(x) = k$, by the definition of $f^*(x)$, we have $\|f(x) - y_k\| < \delta$. Since we have chosen y_k to be in the interior of the cell $\{y : \alpha(y) = \pi(k)\}$, setting δ sufficiently small guarantees that $f(x)$ lies inside the cell $\{y : \alpha(y) = \pi(k)\}$ whenever $x \in \Omega$ and $\alpha^*(x) = k$. That is, $\alpha(f(x)) = \alpha(f^*(x))$ for all $x \in \Omega$. In the unlikely event that $x \notin \Omega$, there would be an encoding error. But as the source is bounded, so is the distortion, by say D_{\max} . Thus the expected distortion conditioned on an encoding error is at most D_{\max} . Since $(1 - P(\Omega))D_{\max}$ can be made less than $\epsilon/4$ by taking δ arbitrarily small, we have

$$D(\alpha \circ f, g \circ \beta) \quad (15)$$

$$= E[d(X, g(\beta(\alpha(f(X)))))] \quad (16)$$

$$\leq P(\Omega)E[d(X, g(\beta(\alpha(f(X))))|\Omega] \quad (17)$$

$$+ (1 - P(\Omega))D_{\max} \quad (18)$$

$$\leq E[d(X, g(\beta(\alpha(f^*(X)))))] + \epsilon/4. \quad (19)$$

We can use a similar argument to show the existence of a post-processor g sufficiently close to g^* . However, if the number of possible reproductions is finite (which is actually implied by our assumption that the codelengths are bounded), then a less sophisticated argument is needed, since then g and g^* need to be close only on a finite set of points. In such case, it is clear that for $\delta > 0$, there exists g such that for all $k \in \mathcal{K}$, $\|g(\beta(\pi(k))) - g^*(\beta(\pi(k)))\| < \delta$. Hence by the continuity of the function $h(\hat{x}) = E[d(X, \hat{x})]$ in \hat{x} , for sufficiently small $\delta > 0$ we have

$$E[d(X, g(\beta(\alpha(f^*(X)))))] \quad (20)$$

$$= E[d(X, g(\beta(\pi(\alpha^*(X)))))] \quad (21)$$

$$\leq E[d(X, g^*(\beta(\pi(\alpha^*(X)))))] + \epsilon/4 \quad (22)$$

$$= E[d(X, \beta^*(\alpha^*(X)))] + \epsilon/4 \quad (23)$$

$$= D(\alpha^*, \beta^*) + \epsilon/4 \quad (24)$$

Together, (19) and (24) result in (13), and thus (11) is proved.

Next we prove (12), by showing that if the approximation is good enough, then the expected rate increases by at most $D(p||q)$, namely

$$R(\alpha \circ f, \gamma) \leq R(\alpha^*, \gamma^*) + D(p||q) + \epsilon. \quad (25)$$

Together, (10) and (25) imply (12).

To show (25), we take a similar strategy to showing (13). First, analogous to (19), we have

$$R(\alpha \circ f, \gamma) \quad (26)$$

$$= E[|\gamma(\alpha(f(X)))|] \quad (27)$$

$$\leq P(\Omega)E[|\gamma(\alpha(f(X)))|\Omega] \quad (28)$$

$$+ (1 - P(\Omega))R_{\max} \quad (29)$$

$$\leq E[|\gamma(\alpha(f^*(X)))|] + \epsilon. \quad (30)$$

Then, analogous to (24), we have

$$E[|\gamma(\alpha(f^*(X)))|] \quad (31)$$

$$= E[|\gamma(\pi(\alpha^*(X)))|] \quad (32)$$

$$= \sum_k P(\{\alpha^*(X) = k\})|\gamma(\pi(k))| \quad (33)$$

$$= -\sum_k p(k) \log_2 q(k) \quad (34)$$

$$= H(p) + D(p||q) \quad (35)$$

$$\leq R(\alpha^*, \gamma^*) + D(p||q), \quad (36)$$

where $p(k) = P(\{\alpha^*(X) = k\})$ and $q(k) = 2^{-|\gamma(\pi(k))|}$. We have also used expressions for the entropy

$$H(p) = -\sum_k p(k) \log_2 p(k) \quad (37)$$

and the Kullback-Leibler divergence

$$D(p||q) = \sum_k p(k) \log_2 \frac{p(k)}{q(k)}. \quad (38)$$

Together, (30) and (36) result in (25), and thus (12) is proved. \square

Note that the permutation was not needed anywhere in the proof. However we get it for free since the y_k s are arbitrary. Moreover, we are now able to optimize over the permutation, to better approximate γ^* with $\gamma \circ \pi$. We now show:

Proposition 3. *For any given codec $(\alpha^*, \beta^*, \gamma \circ \pi)$, the minimum rate $R(\alpha^*, \gamma \circ \pi)$ is achieved when the codelengths $|\gamma(\pi(k))|$ have the same order as $-\log_2 P(\{\alpha^*(X) = k\})$.*

Proof. An expression for the rate is

$$R(\alpha^*, \gamma \circ \pi) = \sum_k P(\{\alpha^*(X) = k\})|\gamma(\pi(k))|. \quad (39)$$

Thus if there exist k_1 and k_2 for which $P(\{\alpha^*(X) = k_1\}) > P(\{\alpha^*(X) = k_2\})$ but $|\gamma(\pi(k_1))| > |\gamma(\pi(k_2))|$, then the rate (39) can be strictly reduced by swapping $\gamma(\pi(k_1))$ and $\gamma(\pi(k_2))$, so that $|\gamma(\pi(k_1))| < |\gamma(\pi(k_2))|$. \square

Note that since γ^* minimizes the rate, the sequences $|\gamma^*(k)|$, $-\log_2 P(\{\alpha^*(X) = k\})$, and $|\gamma(\pi(k))|$ (the latter with a rate-minimizing permutation) all have the same order, up to ties.

Finally, in this section we present an algorithm for learning an optimal Codelength Constrained Vector Quantizer (CCVQ) from data. A CCVQ $(\alpha^*, \beta^*, \gamma^{(0)} \circ \pi)$ comprises an encoder $\alpha : \mathbb{R}^n \rightarrow \mathcal{K}$, a decoder $\beta : \mathcal{K} \rightarrow \mathbb{R}^n$, and a lossless codebook $\gamma : \mathcal{K} \rightarrow \{0, 1\}^*$ minimizing the Lagrangian functional

$$J_\lambda(\alpha, \beta, \gamma) = D((\alpha, \beta)) + \lambda R(\alpha, \gamma) \quad (40)$$

subject to $\gamma = \gamma^{(0)} \circ \pi$ being a reordering of a given invertible lossless codebook $\gamma^{(0)}$. Our CCVQ design algorithm is similar to the Entropy Constrained Vector Quantization (ECVQ) design algorithm of [60], except that instead of assigning a codelength $-\log_2 P(\{\alpha(X) = k\})$ to index k , it must re-use one of the existing codelengths $|\gamma^{(0)}(k)|$.

First some notation. For any set of values $v(k)$ indexed by $k \in \mathcal{K}$, let $\mathbf{k} = \text{argsort}(\{v(k) : k \in \mathcal{K}\})$ denote a list of indices such that the i th element of the list, \mathbf{k}_i , is the index $k \in \mathcal{K}$ of the i th element of the set $\{v(k) : k \in \mathcal{K}\}$ when sorted smallest to largest, with ties broken arbitrarily. The algorithm is shown in Alg. 1.

Algorithm 1 Codelength Constrained Vector Quantization

Input: distribution P , distortion measure d , Lagrange multiplier λ , convergence threshold ϵ , index set \mathcal{K} , decoder $\beta^{(0)} : \mathcal{K} \rightarrow \mathbb{R}^n$, lossless codebook $\gamma^{(0)} : \mathcal{K} \rightarrow \{0, 1\}^*$

- 1: $t = 0, J^{(0)} = \infty, \mathbf{k}^{(0)} = \text{argsort}(\{|\gamma^{(0)}(k)| : k \in \mathcal{K}\})$
- 2: $\forall x: \alpha^{(t+1)}(x) = \text{argmin}_{k \in \mathcal{K}} d(x, \beta^{(t)}(k)) + \lambda |\gamma^{(t)}(k)|$
- 3: $\mathbf{k}^{(t+1)} = \text{argsort}(\{-\log P(\{\alpha^{(t+1)}(X) = k\}) : k \in \mathcal{K}\})$
- 4: $\forall i: \gamma^{(t+1)}(\mathbf{k}_i^{(t+1)}) = \gamma^{(0)}(\mathbf{k}_i^{(0)})$
- 5: $\forall k: \beta^{(t+1)}(k) = \text{argmin}_{\hat{x}} E[d(X, \hat{x}) | \alpha^{(t+1)}(X) = k]$
- 6: $D^{(t+1)} = E[d(X, \beta^{(t+1)}(\alpha^{(t+1)}(X)))]$
- 7: $R^{(t+1)} = E[|\gamma^{(t+1)}(\alpha^{(t+1)}(X))|]$
- 8: $J^{(t+1)} = D^{(t+1)} + \lambda R^{(t+1)}$
- 9: **if** $(J^{(t)} - J^{(t+1)})/J^{(t+1)} > \epsilon$ **then** $t \leftarrow t + 1$ & go to 2
- 10: **end if**

Output: encoder $\alpha^{(t+1)}$, decoder $\beta^{(t+1)}$ and lossless codebook $\gamma^{(t+1)}$ minimizing Lagrangian functional $J_\lambda(\alpha, \beta, \gamma)$ subject to γ being a reordering of $\gamma^{(0)}$



Fig. S7. Comparison of sandwich and HEVC clips at lower rates for the scenario in Fig. 21. As the rate is lowered both the sandwich and HEVC clips have lower LPIPS-PSNR and visual quality. At the same LPIPS quality, it is still difficult to have a firm preference between them. In the top row rate is approximately half that of Fig. 21. Sandwich is still better by $\sim 20\%$ in rate. The bottom row shows the low quality regime. Both clips lose texture detail but HEVC seems to have more artifacts.



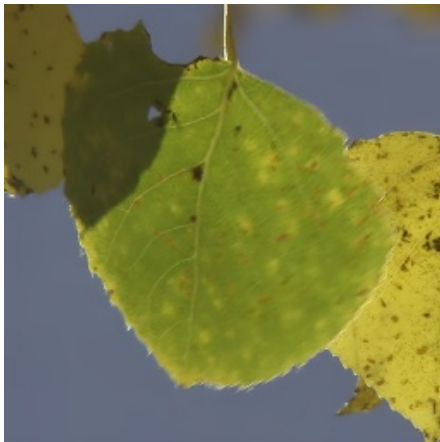
(a) Sandwich (37.3 dB, 0.30 bpp)



(b) HEVC (37.4 dB, 0.48 bpp)



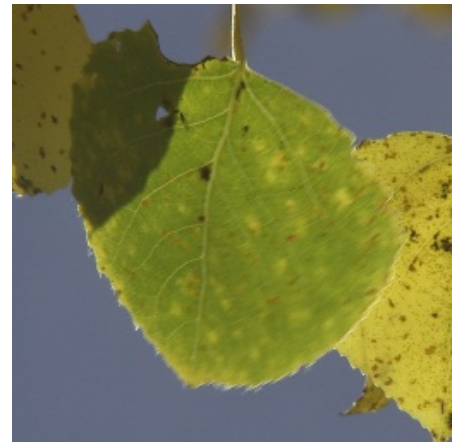
(c) Original



(a) Sandwich (37.8 dB, 0.17 bpp)



(b) HEVC (38.0 dB, 0.22 bpp)



(c) Original



(a) Sandwich (38.5 dB, 0.14 bpp)



(b) HEVC (38.4 dB, 0.15 bpp)



(c) Original

Fig. S8. Qualifying the sandwich rate gains at same LPIPS quality. Top row: On video clips dense with high frequencies and textures the sandwich obtains the most significant gains ($\sim 37.6\%$ for this clip). Middle row: On clips with lesser but still significant high-frequency content gains are reduced but remain significant ($\sim 22.6\%$). Bottom row: On clips showing smooth and blurry regions gains are further reduced ($\sim 5.12\%$).