Deep Perceptual Preprocessing for Video Coding

Aaron Chadha Yiannis Andreopoulos
iSIZE, isize.co

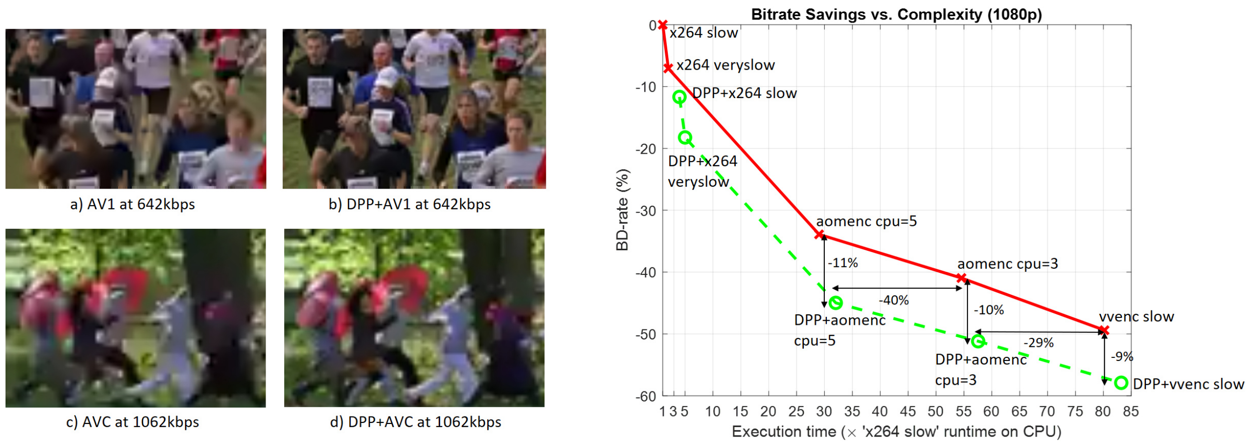


Figure 1: **Left:** Frame segments of encoder vs. DPP+encoder at the same bitrate. **Right:** Bjontegaard delta-rate vs. runtime (in multiples of x264 slow preset runtime on CPU) for codec only (red) and DPP+codec (green). More negative BD-rates correspond to higher average bitrate savings for the same visual quality. x264: AVC/H.264, aomenc: AV1, vvenc: VVC/H.266).

Abstract

We introduce the concept of rate-aware deep perceptual preprocessing (DPP) for video encoding. DPP makes a single pass over each input frame in order to enhance its visual quality when the video is to be compressed with any codec at any bitrate. The resulting bitstreams can be decoded and displayed at the client side without any post-processing component. DPP comprises a convolutional neural network that is trained via a composite set of loss functions that incorporates: (i) a perceptual loss based on a trained no-reference image quality assessment model, (ii) a reference-based fidelity loss expressing L1 and structural similarity aspects, (iii) a motion-based rate loss via block-based transform, quantization and entropy estimates that converts the essential components of standard hybrid video encoder designs into a trainable framework. Extensive testing using multiple quality metrics and AVC, AV1 and VVC encoders shows that DPP+encoder reduces, on average, the bitrate of the corresponding encoder by 11%. This marks the first time a server-side neural processing component achieves such savings over the state-of-the-art in video coding.

1. Introduction

Streaming high-resolution video comes with an inevitable trade-off between available bandwidth and visual quality. In recent years, many video compression standards have been developed, such as Advanced Video Coding (AVC) and AOMedia Video 1 (AV1), which offer a number of advanced coding and prediction tools for efficient video encoding and transmission. While these codecs are widely deployed in industry, with AVC still accounting for the largest share of video streaming volume worldwide, the encoding tools are handcrafted and not entirely data dependent. This has led to an increased interest in learned video compression methods [24, 13, 11, 35], which claim to offer better encoding efficiency by training deep neural networks to improve the rate-distortion performance. However, these methods come with their own pitfalls; typically they require bespoke encoder, bitstream and decoder components for end-to-end optimization. The decoder is typically computationally heavy and not viable for deployment on CPU-based commodity devices such as mobile phones. Additionally, work in learned video compression [24, 45] tends to be benchmarked against codecs with limited tools enabled: 'very fast' preset, low-latency mode and GOP sizes of 10 frames. It is unclear if learned video compression methods outperform standards under their more advanced (and most widely used) encoding settings.

In this work we aim to bridge the gap between the data adaptivity and scalability of learned compression methods and the performance and off-the-shelf decoding support offered by standard codec implementations. To this end, our proposed deep perceptual preprocessor (DPP) simply prepends any standard video codec at inference, without requiring any bespoke encoder of decoder component. The key aspect of our proposal is that it offers rate-aware perceptual improvement by encapsulating both perceptual and fidelity losses, as well as a motion-based rate loss that encapsulates the effect of motion-compensated prediction and entropy coding. In addition, our trained DPP models require a single pass over the input and *all* encodings with *different* standards-based encoders at *various bitrates and resolutions* can be subsequently applied to the DPP output. Experiments versus state-of-the-art AVC, AV1 and Versatile Video Coding (VVC) [14] encoders show that DPP allows for 11% average reduction in bitrate without requiring changes in encoding, streaming, or video decoding.

We summarize our contributions as follows:

1. We propose a deep perceptual preprocessor (DPP) that preprocesses the input content prior to passing it to any standard video codec, such as AVC, AV1 or VVC.
2. We train the DPP in an end-to-end manner by virtualizing key components of a standard codec with differentiable approximations. We balance between perception and distortion by using an array of no-reference and reference based loss functions.
3. We test our models under the most stringent testing conditions: multi-resolution, multi-QP, convex-hull optimization per clip and high-performance AVC, AV1 and VVC presets used extensively by Netflix, Facebook, Intel in several benchmark papers [19, 21, 20].

Visual comparisons of encoder versus DPP+encoder outputs are shown in Fig. 1 (left), illustrating the visual quality improvement that can be achieved at the same bitrate. Fig. 1 (right) illustrates how DPP is able to offer consistent bitrate savings across three video coding standards of increasing sophistication and complexity, while its runtime overhead diminishes in comparison to the encoding runtime.

2. Related Work

2.1. Compression

Recent work in learned image [2, 3, 34, 39] or video [24, 13, 11, 35] compression tends to replace the entirety of a standard transform coding pipeline with neural networks. That is, a neural network-based encoder learns to transform an image or video $\boldsymbol{x}$ into a latent vector $\boldsymbol{y}$. The latent vector is quantized, yielding a discrete valued representation $\hat{\boldsymbol{y}}$, upon which rate is minimized via differential entropy computation:

$$R = -\mathbb{E}_{\hat{\boldsymbol{y}}} \log_2 p(\hat{\boldsymbol{y}}) \tag{1}$$

Given that quantization and prior density $p(\hat{\boldsymbol{y}})$ estimation for entropy computation are non-differentiable operations [2, 39], these are instead represented with continuous approximations. The reconstructed image or video $\hat{\boldsymbol{x}}$ can thus be generated from $\hat{\boldsymbol{y}}$ with a neural-network based decoder. The error between the reconstructed input $\hat{\boldsymbol{x}}$ and original input $\boldsymbol{x}$ can be minimized via a distortion measure Δ, such as mean squared error (MSE) or mean absolute error (MAE):

$$D = \mathbb{E}_{\boldsymbol{x},\hat{\boldsymbol{x}}} \Delta(\boldsymbol{x}, \hat{\boldsymbol{x}}) \tag{2}$$

The encoder and decoder thus constitute an (variational) autoencoder framework [2, 3, 11, 16], and this framework is trained end-to-end to jointly optimize rate and distortion with loss $\mathcal{L} = D + \lambda R$, where λ is the Lagrange multiplier that controls the rate-distortion tradeoff [30]. In the case where the prior density model is fully factorized, statistical dependencies between elements of $\hat{\boldsymbol{z}}$ can be modelled with a (scale) hyperprior [3, 11]; however, any additional encoding bits must be transmitted as side information.

Contrary to recent methods in learned compression, standard image or video codecs typically adopt orthogonal linear transforms to the frequency domain, where the data is decorrelated and easier to compress. While the transform coefficients are not necessarily data adaptive and can exhibit strong joint dependencies [36, 37], the parameters are exposed and can be finely tuned. While learned video compression has shown some promise for high-bitrate low-delay video compression [24, 2, 3, 13, 35], standard codecs like AVC and HEVC surpass all current methods in learned video compression in terms of standard metrics like SSIM and VMAF when the former are used with all their advanced prediction and entropy coding tools enabled [12]. In addition, more advanced encoder designs of the AOMedia AV1 [15] and MPEG/ITU-T Versatile Video Coding (VVC) standards [40] now include neural components for optimized encoding tool selection [15]. Such standards allow for decoders on CPU-based commodity devices like tablets and mobile phones and there is no need for bespoke encoder or decoder components that require joint optimization, as in recent proposals [10, 1].

2.2. Metrics

Performance of compression methods is typically evaluated by plotting rate-distortion curves. Rate is measured in bits per pixel (bpp) or bits-per-second (bps) for video. In recent work, distortion is typically evaluated in terms of

PSNR or SSIM. However, while these metrics are viable options for measuring reconstruction error from the source, they do not capture perceptual quality of the content. Perceptual quality is instead captured by the divergence between the distribution of reconstructed images $p(\hat{\boldsymbol{x}})$ and original images $p(\boldsymbol{x})$. Blau *et al.* [8] mathematically proved the existence of a perception-distortion bound where distortion *must* be traded off with perceptual quality or vice versa. This work was extended further to incorporate rate, where it was derived that, in order to improve perceptual quality, either rate or distortion must be increased [9]. Indeed, for constant rate, distortion must be increased to increase perceptual quality and this tradeoff is strengthened at low rates. Furthermore, perfect perceptual quality cannot be achieved by only optimizing a distortion measure. However, the tradeoff between perception and distortion for constant rate can be weakened for perceptually-oriented distortion measures that capture more semantic similarities.

Given the above, we consider other metrics for evaluating our method, beyond SSIM and PSNR. Notably, VMAF is a perceptually-oriented full-reference (FR) distortion metric, which has been developed and is commercially adopted by Netflix, Facebook, Intel, AOMedia standardization, and several others for codec evaluation [19, 21, 20, 32, 29, 15] and A/B experimentation [20]. VMAF has two primary components: visual information fidelity (VIF) and detail loss metric (DLM), and their respective scores are fused into a single prediction with support vector regression (SVR). Multiple independent studies have shown that VMAF is significantly more correlated to human opinion scores than SSIM and PSNR [31, 4, 47].

Recently, a more compression-oriented variant of VMAF, VMAF_NEG [20], has been proposed by Netflix for isolating compression artifacts from general perceptual quality enhancement (e.g., contrast enhancement). Essentially, VMAF_NEG is derived by clipping the local gain terms in VIF and DLM to 1.0, thus penalizing linear enhancement operations. In this paper, we present results in terms of VMAF, VMAF_NEG and SSIM, to demonstrate how our method traverses the perception-distortion space.

3. Deep Perceptual Preprocessor

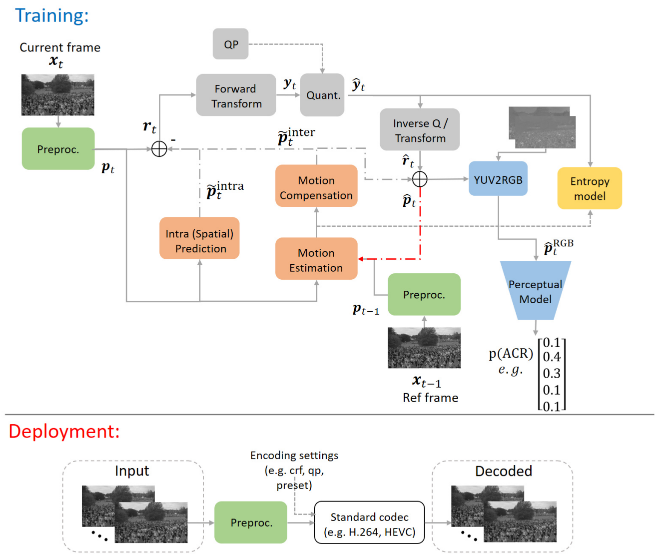
3.1. Overview of Proposed Method

In this section, we describe our deep perceptual preprocessing (DPP) framework for video preprocessing. Essentially, the objective of our preprocessing framework is to provide a perceptually optimized and rate-controlled representation of the decoded input frame via a learnable preprocessing approach. On one hand, the preprocessing must have some level of encoder-awareness such that it can adapt to visual distortions induced by changing standard codec settings such as quantization parameter (QP) and constant rate value (CRF). On the other hand, in order to maintain a single-pass preprocessing and to avoid training a preprocessing model for every single codec and configuration, the preprocessing must have a marginalized response over the codec parameter space. To this end, we propose to model or 'virtualize' the basic building blocks of a standard video coding pipeline, such that we can approximate the rate-distortion behavior over standard video codecs. The core codec components we model are inter/intra prediction, adaptive macroblock selection, spatial frequency transform and quantization. This virtual codec is appended to our preprocessing neural network and the resulting DPP framework is trained *end-to-end* with our proposed loss formulations. In this way, we perform perceptual and codec-oriented rate-distortion optimization over the preprocessing network parameters. Notably, in order to aid with marginalization, we also expose parameters such as QP, which can be adjusted during training. During inference/deployment, the virtual encoder is removed and replaced with a standard codec, such as an MPEG or AOMedia encoder.

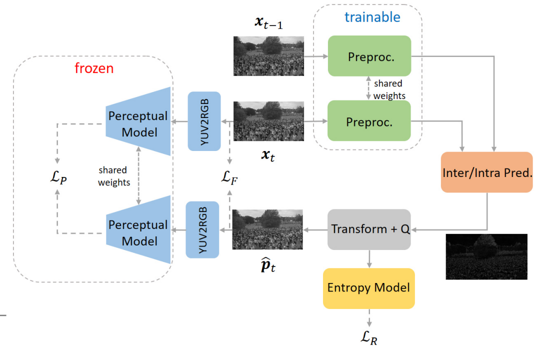
The training and deployment frameworks are illustrated in Figure 2a. Each color outlines a different component in the training framework. For a given video sequence $\mathbf{V} = \{\boldsymbol{x}_1 \dots \boldsymbol{x}_t, \boldsymbol{x}_{t+1} \dots \boldsymbol{x}_N\}$ with N frames, the green blocks represent the preprocessing network that maps input video frame $\boldsymbol{x}_t$ at time t to preprocessed frame $\boldsymbol{p}_t$. The orange blocks represent the components for inter (motion estimation + compensation) and intra prediction, which output a predicted frame $\tilde{\boldsymbol{p}}_t$ and residual frame $\boldsymbol{r}_t$ by performing block matching between the current and reference frames. Importantly, in this paper we focus on an open loop codec implementation for inter prediction and exclude the red arrow in the figure. The grey blocks represent the spatial transform and quantization components for encoding and compressing the residual. The residual frame is transformed to the frequency domain output $\boldsymbol{y}_t$ and quantized to $\hat{\boldsymbol{y}}_t$, with the quantization level controlled by the quantization parameter (QP). We model the rate of $\hat{\boldsymbol{y}}_t$ with an entropy model, as represented with the yellow block, as this is what a standard encoder would losslessly compact into the compressed bitstream. The blue blocks represent YUV to RGB conversion and the perceptual model that we use collectively to quantify perceptual quality, based on mean opinion scores (MOS). These components will allow us to train the preprocessing network to enhance the perceptual quality of reconstructed frame $\hat{\boldsymbol{p}}_t$.

3.2. Learnable Preprocessing

The input video frames are first processed individually by a preprocessing block, represented in green in Figures 2a and 2b. The preprocessing block $F(\boldsymbol{x}; \Theta)$ comprises a pixel-to-pixel mapping F, with associated parameters Θ. For efficient deployment, preprocessing only processes the



(a) Deep perceptual preprocessor (DPP) framework



(b) Open loop schematic for DPP

Figure 2: (a): Deep perceptual preprocessor framework for training perceptually-enhanced & rate-controlled representation of input frames via a learnable preprocessing. Dashed arrows represent optional components. (b): Schematic showing the perceptual preprocessor training framework in open loop configuration with loss functions.

luminance (Y) channel only, since it contains all of the frame's structural information and is the main contributor to perceptual sharpness and bitrate, which constitute our main objectives for optimization. Specifically, for input frame $\boldsymbol{x} \in \mathbb{R}^{H\times W}$ scaled to range $[0, 1]$ and modelled representation $\hat{\boldsymbol{p}}$, the intention is to optimize parameters Θ, in order to achieve a balance on $\hat{\boldsymbol{p}}$ between the perceptual enhancement, rate control and fidelity to $\boldsymbol{x}$. The mapping F is implemented as a convolutional neural network (CNN) with single-frame latency (assuming the supporting hardware can carry out the CNN inference fast enough). In order to reduce the network complexity while allowing for larger receptive field sizes and maintaining translational equivariance, we utilize dilated convolutions [46] with varying dilation rates per layer. The neural network weights constitute the parameters Θ that we intend to optimize for perceptual quality, rate and distortion in our training framework.

3.3. Inter and Intra Prediction

The preprocessing network maps current video frame $\boldsymbol{x}_t$ at time step to t to $\boldsymbol{p}_t$. The next step is to generate the residual frame $\boldsymbol{r}_t$ via intra or inter prediction. A standard video codec such as H.264/AVC adaptively divides the frame into variable-sized macroblock partitions and sub-partitions, typically varying from 16×16 to 4×4. Let us first assume a fixed block size. Under this assumption, the preprocessed frame $\boldsymbol{p}_t$ is first divided into a set of blocks of the fixed size $K \times K$. For a block in the current frame centered on the pixel location $(n_1, n_2) \in [(0, 0), (H-1, W-1)]$, a local search space centered on (n_1, n_2) and of size $M \times M$ is extracted from the reference frame. A similarity criterion is used to find the best matching block of size $K \times K$ to the current frame block within the local search space. For inter prediction, the local search space is extracted from the previous frame, $\boldsymbol{p}_{t-1}$. The similarity criterion ϵ can thus be expressed at (n_1, n_2) as:

$$\epsilon(m_1, m_2) \triangleq \sum_{(k_1,k_2)} d(\boldsymbol{p}_t(n_1 + k_1, n_2 + k_2), \boldsymbol{p}_{t-1}(n_1 + k_1 + m_1, n_2 + k_2 + m_2)) \quad (3)$$

where the coordinates $(k_1, k_2) \in [(0, K-1), (0, K-1)]$ shift the pixel location within a $K \times K$ block and $(m_1, m_2) \in [(-\frac{M}{2}, -\frac{M}{2}), (\frac{M}{2}, \frac{M}{2})]$ represent the block displacement within the local search space of the reference frame. d represents the similarity measure, which in this paper is set to mean absolute error (MAE), given its better handling of outliers than mean squared error (MSE). Importantly, the operation in (3) can be easily vectorized, which enables efficient end-to-end training on GPUs (at the cost of higher memory allocation). Then, for the given current frame block, the optimal block displacement $\boldsymbol{m} = (m_1^*, m_2^*)^{\mathrm{T}}$ in the reference frame is given as:

$$(m_1^*, m_2^*) = \underset{(m_1,m_2)}{\arg\min}(\epsilon(m_1, m_2)) \quad (4)$$

The displacement or motion vector $\boldsymbol{m}^* = (m_1^*, m_2^*)^{\mathrm{T}}$ is encoded for each block in the current frame. However, the $\arg\min$ in (4) has zero gradients almost everywhere with respect to the input and therefore is not differentiable. This poses a problem if we wish to optimize the DPP with end-to-end backpropagation from the reconstructed frame $\hat{\boldsymbol{p}}_t$ back to the input frame $\boldsymbol{x}_t$. In order to resolve this, we first express (4) in terms of a one-hot matrix, which we denote as $\mathbf{1}_{\arg\min_{(\boldsymbol{m})}(\epsilon)}$, where the matrix is 1 at index (m_1^*, m_2^*) and 0 for all other (m_1, m_2). We approximate the argmin operation by using a straight-through estimator [5]. Our approach is analogous to gumbel-softmax [17] except that we are not sampling over a discrete distribution but deterministically extracting the optimal block based on ϵ. The predicted frame $\tilde{\boldsymbol{p}}_t^{\mathrm{inter}}$ is then configured as: $\tilde{\boldsymbol{p}}_t^{\mathrm{inter}}(n_1 + k_1, n_2 + k_2) = \sum_{(m_1,m_2)} \mathbf{1}_{(\boldsymbol{m}^*)}(m_1, m_2).\boldsymbol{p}_{t-1}(n_1 + k_1 + m_1, n_2 + k_2 + m_2)$ and the residual frame $\boldsymbol{r}_t$ is simply equal to the difference between the predicted frame and current frame: $\boldsymbol{r}_t = \boldsymbol{p}_t - \tilde{\boldsymbol{p}}_t^{\mathrm{inter}}$.

For intra prediction, we follow a similar approach for generating $\tilde{\boldsymbol{p}}_t^{\mathrm{intra}}$, except the reference frame from which we extract the local search space is from the current frame $\boldsymbol{p}_t$ itself (but masking the block being queried and only searching in the causal neighborhood around the queried block). In this way, we are able to emulate all translational intra prediction modes.

3.4. Transform and Quantization

The residual frames $\boldsymbol{r}_t$ are transformed in our framework into the frequency domain for further energy compaction, akin to a standard video codec. The forward transform is typically a two-dimensional discrete transform (DCT) followed by quantization. In this paper, we opt for the 4×4 core and scale transforms of the integer DCT defined in the H.264/AVC standard [27], after rescaling $\boldsymbol{r}_t$ between [0,255]. The transformed and scaled frame $\boldsymbol{y}_t$ is then quantized by dividing by a quantization value Q_{step} and rounding, with Q_{step} being randomly selected during training from a range of values. We manually assign the first 6 values of Q_{step} based on the equivalent values for AVC QP in the range [0,5]. We can then draw a direct equivalence between Q_{step} and the QP setting used in AVC encoding [33]. We denote the quantized frame as $\hat{\boldsymbol{y}}_t$. We further note that the rounding operation in quantization is non-differentiable – we thus approximate rounding with additive uniform noise during training (i.e. $\hat{\boldsymbol{y}}_t = \frac{\boldsymbol{y}_t}{Q_{\mathrm{step}}} + \Delta\boldsymbol{y}_t$, where $\Delta\boldsymbol{y}_t$ is additive i.i.d uniform noise with support of width 1). In a standard video coding pipeline, $\hat{\boldsymbol{y}}_t$ is the representation that would be encoded to bits with an entropy coder such as CAVLC or CABAC [28]. The quantization and forward transform can then be inverted by multiplying by Q_{step} and taking the inverse integer DCT, thus producing the reconstructed residual $\hat{\boldsymbol{r}}_t$. The reconstructed frame $\hat{\boldsymbol{p}}_t$ is equal to $\tilde{\boldsymbol{p}}_t + \hat{\boldsymbol{r}}_t$.

3.5. Entropy Model

Given that we aim to optimize our preprocessing on rate, we must minimize the number of bits required to encode the DCT transformed and quantized frame $\hat{\boldsymbol{y}}_t$. This can be estimated by computing the entropy as in (1). However, as discussed, the prior density $p(\hat{\boldsymbol{y}}_t)$ must be estimated with a continuously differentiable approximation, such that we can compute the number of bits to encode the DCT subbands in a differentiable manner. To this end, we can model $p(\hat{\boldsymbol{y}}_t)$ as a factorized prior. The disadvantage of assuming a factorized prior on $\hat{\boldsymbol{y}}$ is that it does not account for the strong non-linear dependencies between subband coefficients [36, 25, 41]. Rather than extending the factorized prior with a hyperprior [3], which would require additional training and deviate further from standard codec operation, we propose a simple spatial divisive normalization which has been shown to decorrelate DCT domain coefficients per sub-band [26]. We denote the divisively normalized coefficients as $z_{n,s,t}$, where index n runs per subband over all spatial coordinates.

In this way, we can assume a factorized prior $p(\boldsymbol{z})$ on $\boldsymbol{z}$ instead of $\hat{\boldsymbol{y}}$:

$$p(\boldsymbol{z}_t; \Phi) = \prod_{n,s} p(z_{n,s,t}; \Phi^{(s)}) \tag{5}$$

In other words, we assume an independent univariate density model for each subband, parameterized by $\Phi^{(s)}$, but that all spatial dimensions are i.i.d.. Each subband model is learned with the non-parametric implementation defined by Balle *et al.* [3]. During end-to-end training, we can reconstruct $\hat{\boldsymbol{y}}$ from $\boldsymbol{z}$ by simply taking the inverse transform.

3.6. Perceptual Model

We aim to perceptually enhance our decoded input frame representations $\hat{\boldsymbol{p}}_t$, under the rate and distortion constraints introduced by lossy compression. Rather than train full-reference (FR) perceptual model that would transform pairwise distortion between $\hat{\boldsymbol{p}}_t$ and $\boldsymbol{x}_t$ into a MOS score, we opt for a no-reference (NR) perceptual model that can better encapsulate deviations from natural scene statistics to assess perceptual quality of $\hat{\boldsymbol{p}}_t$. The requirement of NR models to assess perceptual quality has been extensively discussed by Blau *et al.* [8, 9]. Given that we do not have access to MOS scores for preprocessed frames, we must first pretrain the perceptual model. The trained perceptual model is thus frozen in the DPP framework and used to derive the perceptual loss $\mathcal{L}_P$, as illustrated in Figure 2b. Our architecture is a directed acyclic graph (DAG) variant of NIMA [38]. Essentially, we fine-tune a VGG-16 model that has been pre-trained on ImageNet. The fully connected layers are removed and replaced with global average pooling and single fully connected layer with 5 neurons. A softmax function maps the output to a distribution over human ratings, or ACR distribution, ranging from from poor (1) to excellent (5). To give the output layer access to multi-scale and multi-semantic representations of the input, we also global average pool intermediate layer activations and concatenate the pooled activations over layers. The model is thus trained to minimize the total variation distance between predicted and reference human rating distributions. We note that given that our perceptual model is trained on human-rated RGB images, it is necessary in our perceptual preprocessing framework to first convert the luminance frame $\hat{\boldsymbol{p}}_t$ to RGB frame $\hat{\boldsymbol{p}}_t^{\mathrm{RGB}}$. We perform a transform from YUV to RGB space by first concatenating $\hat{\boldsymbol{p}}_t$ with the lossless U and V components of the RGB input, $\boldsymbol{x}_t^{\mathrm{RGB}}$.

3.7. Loss Functions

Our overall objective is to train our preprocessing $F(\boldsymbol{x}_t;\Theta)$ to perform perceptually-oriented rate-distortion optimization on the decoded frame representations $\hat{\boldsymbol{p}}_t$ relative to the input video frames $\boldsymbol{x}_t$. Assuming the domain shift between our virtual codec and standard video codec is marginal, this should equate to optimizing the rate and distortion of the decoded frames during deployment with a standard video codec. To this end, we train the CNN of the preprocessor end-to-end with the building blocks of our DPP framework and a perceptual loss ($\mathcal{L}_{\mathrm{P}}$), rate loss ($\mathcal{L}_{\mathrm{R}}$) and fidelity loss ($\mathcal{L}_{\mathrm{F}}$) (as illustrated in Figure 2b). The overall loss function for training the preprocessing can thus be written as a weighted summation: $\mathcal{L}(\boldsymbol{x}_t,\hat{\boldsymbol{p}}_t;\Theta) = \gamma\mathcal{L}_{\mathrm{P}} + \lambda\mathcal{L}_{\mathrm{R}} + \mathcal{L}_{\mathrm{F}}$, where γ and λ are the perceptual and rate coefficients respectively. It is worth noting that contrary to neural encoders, where changing λ maps to a new rate-distortion point, λ in this case shifts the entire rate-distortion curve mapped over multiple QPs/CRFs - this behavior is illustrated in the ablation study on λ in the supplementary. Given that we marginalize over QP, λ gives the flexibility to explore the entire rate-distortion space.

Fidelity Loss, $\mathcal{L}_{\mathrm{F}}$: In order to ensure a likeness between the input luminance frame $\boldsymbol{x}_t$ and the perceptually enhanced and rate constrained decoded frame representation $\hat{\boldsymbol{p}}_t$, we train the preprocessing with a combination of fidelity losses. As discussed by Zhao *et al.* [48], the L_1 distance is good for preserving luminance, whereas multiscale structural similarity (MS-SSIM) [42] is better at preserving contrast in high frequency regions. Our fidelity loss can thus be written as the summation:

$$\begin{aligned}\mathcal{L}_{\mathrm{F}}(\boldsymbol{x}_t,\hat{\boldsymbol{p}}_t;\Theta) = \mathbb{E}_{\boldsymbol{x}_t,\hat{\boldsymbol{p}}_t}\big[&\alpha\mathcal{L}^{L_1}(\boldsymbol{x}_t,\hat{\boldsymbol{p}}_t;\Theta)\\ &+\beta(1-\mathcal{L}^{\mathrm{MS-SSIM}}(\boldsymbol{x}_t,\hat{\boldsymbol{p}}_t;\Theta)\big]\end{aligned} \tag{6}$$

where $\mathcal{L}^{L_1}(\boldsymbol{x}_t,\hat{\boldsymbol{p}}_t) = |\boldsymbol{x}_t - \hat{\boldsymbol{p}}_t|$ and $\mathcal{L}^{\mathrm{MS-SSIM}}$ represents the MS-SSIM function (as defined by Wang *et al.* [42]), and α and β are hyperparameters which control the weighting on structural versus luminance preservation.

Rate Loss, $\mathcal{L}_{\mathrm{R}}$: The virtual codec rate loss $\mathcal{L}_{\mathrm{R}_s}$ per DCT sub-band s is defined on the divisively normalized transform coefficients $\boldsymbol{z}_t$:

$$\mathcal{L}_{\mathrm{R}_s}(\boldsymbol{z}_t;\Theta,\Phi) = -\mathbb{E}_{\boldsymbol{z}_t}\sum_n(\log_2(p(z_{n,s,t};\Phi^{(s)})) \tag{7}$$

where n runs over all spatial coordinates of each sub-band. The final rate loss is simply the summation over all sub-bands: $\mathcal{L}_{\mathrm{R}} = \sum_{s=1}^{S}\mathcal{L}_{\mathrm{R}_s}$, where $S = 16$ for a 4×4 DCT. The rate loss represents an approximation (upper bound) to the actual rate required to encode the preprocessed frames.

Perceptual Loss, $\mathcal{L}_{\mathrm{P}}$: We quantify perceptual quality with our perceptual model P, which is pre-trained and frozen during the DPP training. Essentially, we aim to maximize the mean opinion scores (MOS) of our decoded RGB frame representations $\hat{\boldsymbol{p}}_t^{\mathrm{RGB}}$, independent of the reference frame $\boldsymbol{x}_t^{\mathrm{RGB}}$, but derived on the natural scene statistics (NSS) learned from training the perceptual model on a corpus of natural images. To this end, we minimize:

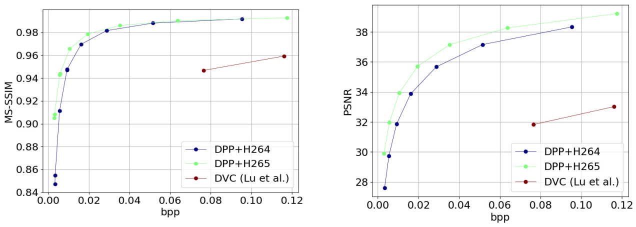
$$\mathcal{L}_{\mathrm{P}}(\hat{\boldsymbol{p}}_t;\Theta) = -\mathbb{E}_{\hat{\boldsymbol{p}}_t}\sum_{i=1}^{5} i(P(\hat{\boldsymbol{p}}_t^{\mathrm{RGB}})_i) \tag{8}$$

where the inner summation represents the predicted MOS score, as the mean over the predicted ACR distributions.

4. Experimental Results

4.1. Implementation Details

The perceptual model P is first trained on Koniq-10k no-reference IQA dataset [23] using stochastic gradient descent with momentum set to 0.9 and an initial learning rate of 1×10^{-3}. The perceptual model is then frozen and the deep preprocessing framework is trained on Vimeo-90k dataset [44] in an end-to-end manner, under the open loop configuration illustrated in Figure 2b and loss function as defined in Section 3.7. We train a deep convolutional neural network with curriculum learning [6, 43] and using multi-scale crop sizes. The curriculum is generated via a scoring and pacing function, which map the content type and difficulty. Each convolutional layer is followed by a parametric ReLu activation function. During training we alternate between our inter and intra prediction blocks; we follow a standard encoding pipeline and default to inter prediction only, switching to intra prediction for 1 mini-batch every 100 training iterations (i.e. in correspondence to 1 I-frame every 100 P or B frames). The local search space size M is fixed at 24. The network is trained with Adam optimizer and learning rate is decayed when metrics saturate on the validation dataset. Finally, we follow Zhao *et. al* [48] and fix hyperparameters



(a) MS-SSIM (b) PSNR

Figure 3: Proposed DPP+H264 and DPP+H265 versus DVC [24] on the first 100 frames of HEVC Class B sequences. Points are plotted up to 0.12 bits per pixel (bpp).

α and β to 0.2 and 0.8 respectively. For the core hyperparameters that control the rate-perception-distortion tradeoff, λ and γ, we fix γ to 0.01 and vary $\lambda \in [0.001, 0.01]$. We present an ablation of these parameters in the supplementary material.

At deployment, we only retain the part of the preprocessing that comprises the learned pixel-to-pixel mapping; the virtual codec is replaced with a standard video codec, with the decoded frame perceptually enhanced and at the same or lower bitrate than achievable without any preprocessing. Importantly, we achieved real-time performance for full-HD video (1080p@50fps) on a single NVIDIA Tesla T4 GPU by porting our trained models to OpenCV CUDA primitives and fp16 arithmetic. For CPU execution, by porting our models to OpenVINO and quantizing them to int8, we achieved real time for 1080p@60fps on 12 cores of an Intel Cascade Lake CPU with no detriment in visual quality.

4.2. Experimental Setup for BD-Rate Results

We present a detailed evaluation of different models using standard 1080p XIPH and CDVL sequences[1]. Our anchor encoders comprise AVC/H.264, AV1 and VVC, utilizing the libx264, aomenc and vvenc open implementations of these standards. We deliberately focus on a very-highly optimized encoding setup that is known to outperform all neural or run-of-the-mill proprietary video encoders by a large margin [18, 12, 40, 10]. Our aim is to examine if DPP can push the envelope of what is achievable today under some of the most-advanced encoding conditions used in practice.

Our x264/AVC encoding recipe is: veryslow preset, tune SSIM and multiple CRF values per resolution. Our aomenc AV1 recipe is: two-pass encoding, CPU=5, 'tune SSIM' or 'tune VMAF' preprocessing options, and multiple target bitrates per resolution[2]. Our vvenc recipe used the slow preset and multiple QPs per resolution. All encodings were produced using GOP size of 150 frames (128 for VVC) and for multiple resolutions, ranging from the 1080p original resolution all the way to 144p by using FFmpeg Lanczos downscaling. All lower resolutions are upscaled with FFmpeg bicubic to 1080p prior to quality measurements [22]. All Bjontegaard delta-rates (BD-rates) [7] are produced by first finding the subset of monotonically-increasing bitrate-quality points that are in the convex hull of the quality-bitrate curve, and then using the Netflix libvmaf repository [22] to measure SSIM, VMAF_NEG, VMAF and BD-rates. The convex hull is computed over all resolutions, CRFs/bitrates and multiple rate coefficients λ, such that, per metric, we obtain a single RD-curve for both the codec and our proposed DPP+codec. Full details of this convex hull optimization, along with the utilized encoding recipes can be found in the supplementary.

4.3. Comparison Against Neural Encoders

Before moving to our main results, we present a short comparison against neural encoders, selecting the recently-proposed DVC framework [24] as a representative candidate of the state-of-the-art. Such neural encoders have been shown to outperform AVC and HEVC when the latter are using: no B slices, 'veryfast' preset, low-latency mode (which disables most advanced temporal prediction tools), and very small GOP sizes of 10 or 12 frames. However, they are not able to approach the performance of these hybrid encoders, or indeed that of our framework under the state-of-the-art experimental setup of Section 4.2. This is evident in the example results of Fig. 3, where DVC is very substantially outperformed in terms of bitrate vs. PSNR and MS-SSIM (the metrics used in their work) by both DPP+AVC/H.264 and DPP+HEVC/H.265 under our encoding recipe.

4.4. BD-Rate Results with H.264/AVC and AV1

The results of Fig. 4 and Table 1 and Table 2 show that the average rate saving over VMAF, VMAF_NEG and SSIM for both H.264 and AV1 standards is just above 11%. As expected, our gains are higher on metrics that are increasingly perception-oriented rather than distortion-oriented: on VMAF, our framework offers 18% to 25% saving; on VMAF_NEG, they are between 7% to 11% and on SSIM they are 1% to 3%. This makes the average BD-rate of all three metrics a reliable estimate of the bitrate saving that can be offered in practice, since this average is influenced by performance in both distortion (SSIM) and perception-oriented dimensions (VMAF and VMAF_NEG).

[1]XIPH source material: https://media.xiph.org/video/derf/ and CDVL material: https://www.cdvl.org/. See supplementary results for more details on exact sequences used.

[2]We note that preprocessing techniques such as 'tune VMAF' and 'tune SSIM' operate in-loop, i.e., within a specific encoder. As such, our method can offer gains on top of them.

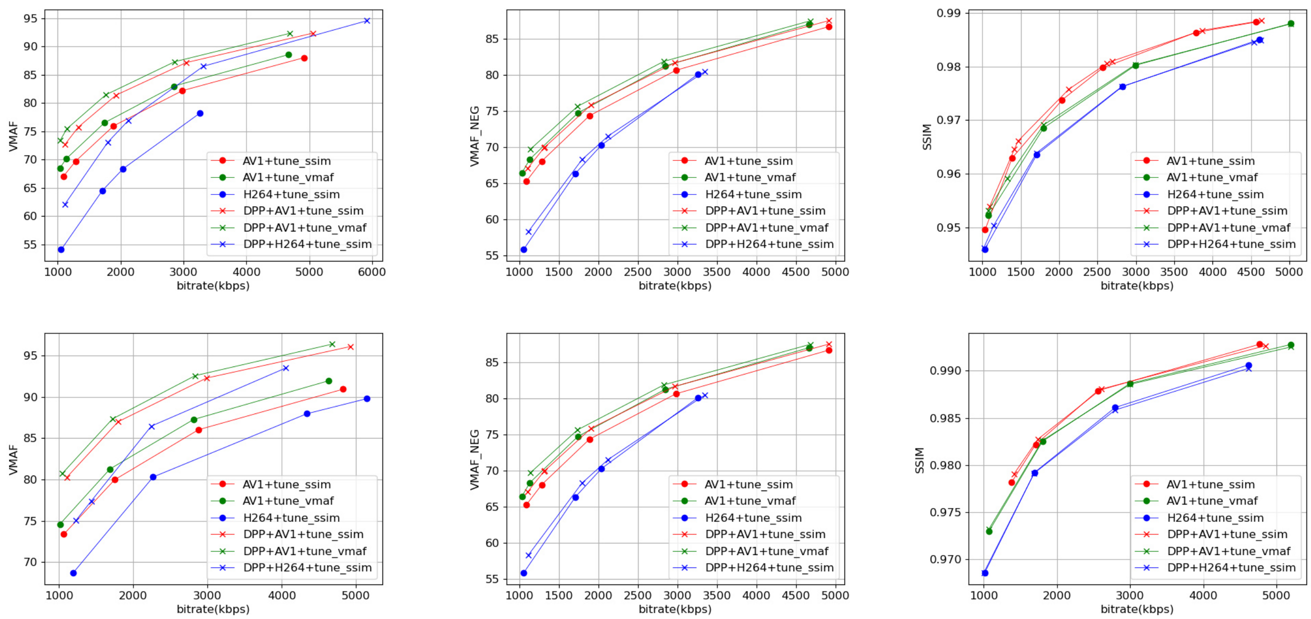


Figure 4: Rate distortion curves for 16 XIPH sequences (top row) and 24 CDVL sequences (bottom row) on VMAF, VMAF_NEG and SSIM respectively. Curves are plotted for the codec and for our proposed DPP+codec. The corresponding BD rates for our method are reported in Tables 1 and 2, respectively, for each dataset.

	VMAF	VMAG_NEG	SSIM
DPP+H264+tune_ssim	-18.57	-11.37	-2.93
DPP+AV1+tune_ssim	-22.03	-10.64	-2.45
DPP+AV1+tune_vmaf	-19.44	-7.98	-2.23
DPP+VVC	-17.08	-4.71	-4.55

Table 1: BD rates on 16 XIPH sequences for DPP+ H264, DPP+AV1 (with perceptual settings tune_ssim and tune_vmaf) and DPP+VVC. More negative=more saving.

	VMAF	VMAG_NEG	SSIM
DPP+H264+tune_ssim	-19.80	-11.41	-2.73
DPP+AV1+tune_ssim	-25.23	-11.41	-2.47
DPP+AV1+tune_vmaf	-24.96	-8.20	-1.20
DPP+VVC	-18.56	-4.93	-2.54

Table 2: BD rates on 24 CDVL sequences for DPP+ H264, DPP+AV1 (with perceptual settings tune_ssim and tune_vmaf) and DPP+VVC. More negative=more saving.

4.5. BD-Rate Results with VVC

We report BD-rate savings for VVC in Table 1 and Table 2. The average saving over all three metrics is 8.7%. The fact that our framework offers consistent savings over vvenc further illustrates the validity of DPP across encoders, encoding recipes, and convex-hull rate-distortion optimized encoding [18], which is summarized in Fig. 1 (right).

5. Conclusion

We propose deep perceptual preprocessing (DPP) as the means of generating a perceptually-enhanced, rate-aware representation of each input frame via a learnable preprocessing framework. DPP models the building blocks of a standard video encoder in order to optimize the proposed preprocessing for rate, distortion and perceptual quality in an end-to-end differentiable manner. At inference, only the preprocessor is deployed to carry out a single pass through each frame prior to any standard encoder. Our framework delivers consistent gains for three quality metrics with different perception-distortion characteristics and for three very different encoders used at their performance limits. It is also easily deployable as it attains real time performance on commodity hardware without requiring any changes in encoding, streaming or video decoding at the client side.

6. Acknowledgements

Y.A. is a Director at iSIZE and also Professor at University College London (UCL), U.K. We thank Ilya Fadeev and Russell Anam for their help with parts of the experiments.

References

[1] Mariana Afonso, Fan Zhang, and David R Bull. Video compression based on spatio-temporal resolution adaptation. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(1):275–280, 2018.

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

[3] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436*, 2018.

[4] Nabajeet Barman, Steven Schmidt, Saman Zadtootaghaj, Maria G Martini, and Sebastian Möller. An evaluation of video quality assessment metrics for passive gaming video streaming. In *Proceedings of the 23rd packet video workshop*, pages 7–12, 2018.

[5] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[6] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[7] Gisle Bjontegaard. Improvements of the BD-PSNR model. In *ITU-T SG16/Q6, 35th VCEG Meeting, Berlin, Germany, July, 2008*, 2008.

[8] Yochai Blau and Tomer Michaeli. The perception-distortion tradeoff. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6228–6237, 2018.

[9] Yochai Blau and Tomer Michaeli. Rethinking lossy compression: The rate-distortion-perception tradeoff. *arXiv preprint arXiv:1901.07821*, 2019.

[10] Eirina Bourtsoulatze, Aaron Chadha, Ilya Fadeev, Vasileios Giotsas, and Yiannis Andreopoulos. Deep video precoding. *IEEE Transactions on Circuits and Systems for Video Technology*, 2019.

[11] Yoojin Choi, Mostafa El-Khamy, and Jungwon Lee. Variable rate deep image compression with a conditional autoencoder. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3146–3154, 2019.

[12] Jan De Cock, Aditya Mavlankar, Anush Moorthy, and Anne Aaron. A large-scale video codec comparison of x264, x265 and libvpx for practical vod applications. In *Applications of Digital Image Processing XXXIX*, volume 9971, page 997116. International Society for Optics and Photonics, 2016.

[13] Abdelaziz Djelouah, Joaquim Campos, Simone Schaub-Meyer, and Christopher Schroers. Neural inter-frame compression for video coding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6421–6429, 2019.

[14] HHI Fraunhofer. Vvenc software repository, 2020. https://github.com/fraunhoferhhi/vvenc.

[15] Adrian Grange, Andrey Norkin, Cheng Chen, Ching-Han Chiang, Debargha Mukherjee, Hui Su, James Bankoski, Jean-Marc Valin, Jingning Han, Luc Trudeau, et al. An overview of core coding tools in the av1 video codec. 2018.

[16] Amirhossein Habibian, Ties van Rozendaal, Jakub M Tomczak, and Taco S Cohen. Video compression with rate-distortion autoencoders. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7033–7042, 2019.

[17] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[18] Ioannis Katsavounidis and Liwei Guo. Video codec comparison using the dynamic optimizer framework. In *Applications of Digital Image Processing XLI*, volume 10752, page 107520Q. International Society for Optics and Photonics, 2018.

[19] Faouzi Kossentini, Hassen Guermazi, Nader Mahdi, Chekib Nouira, Amir Naghdinezhad, Hassene Tmar, Omar Khlif, Phoenix Worth, and Foued Ben Amara. The svt-av1 encoder: overview, features and speed-quality tradeoffs. In *Applications of Digital Image Processing XLIII*, volume 11510, page 1151021. International Society for Optics and Photonics, 2020.

[20] Zhi Li. Video @Scale 2020: VMAF. *At Scale Conference*, 2020. https://atscaleconference.com/videos/video-scale-2020-vmaf/.

[21] Zhi Li, Anne Aaron, Ioannis Katsavounidis, Anush Moorthy, and Megha Manohara. Toward a practical perceptual video quality metric. *The Netflix Tech Blog*, 6, 2016.

[22] Zhi Li, Christos Bampis, Julie Novak, Anne Aaron, Kyle Swanson, Anush Moorthy, and J Cock. Vmaf: The journey continues. *Netflix Technology Blog*, 2018.

[23] Hanhe Lin, Vlad Hosu, and Dietmar Saupe. Koniq-10k: Towards an ecologically valid and large-scale iqa database. *arXiv preprint arXiv:1803.08489*, 2018.

[24] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. Dvc: An end-to-end deep video compression framework. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11006–11015, 2019.

[25] Siwei Lyu and Eero P Simoncelli. Nonlinear image representation using divisive normalization. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[26] Jesús Malo, Irene Epifanio, Rafael Navarro, and Eero P Simoncelli. Nonlinear image representation for efficient perceptual coding. *IEEE Transactions on Image Processing*, 15(1):68–80, 2005.

[27] Henrique S Malvar, Antti Hallapuro, Marta Karczewicz, and Louis Kerofsky. Low-complexity transform and quantization in h. 264/avc. *IEEE Transactions on circuits and systems for video technology*, 13(7):598–603, 2003.

[28] Detlev Marpe, Heiko Schwarz, and Thomas Wiegand. Context-based adaptive binary arithmetic coding in the h. 264/avc video compression standard. *IEEE Transactions on circuits and systems for video technology*, 13(7):620–636, 2003.

[29] Marta Orduna, César Díaz, Lara Muñoz, Pablo Pérez, Ignacio Benito, and Narciso García. Video multimethod assessment fusion (vmaf) on 360vr contents. *IEEE Transactions on Consumer Electronics*, 66(1):22–31, 2019.

[30] Antonio Ortega and Kannan Ramchandran. Rate-distortion methods for image and video compression. *IEEE signal processing magazine*, 15(6):23–50, 1998.

[31] Reza Rassool. Vmaf reproducibility: Validating a perceptual practical video quality metric. In *2017 IEEE international symposium on broadband multimedia systems and broadcasting (BMSB)*, pages 1–2. IEEE, 2017.

[32] Shankar L Regunathan, Haixiong Wang, Yun Zhang, Yu Ryan Liu, David Wolstencroft, Srinath Reddy, Cosmin Stejerean, Sonal Gandhi, Minchuan Chen, Pankaj Sethi, et al. Efficient measurement of quality at scale in facebook video ecosystem. In *Applications of Digital Image Processing XLIII*, volume 11510, page 115100J. International Society for Optics and Photonics, 2020.

[33] Iain E Richardson. *The H. 264 advanced video compression standard*. John Wiley & Sons, 2011.

[34] Oren Rippel and Lubomir Bourdev. Real-time adaptive image compression. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2922–2930. JMLR. org, 2017.

[35] Oren Rippel, Sanjay Nair, Carissa Lew, Steve Branson, Alexander G Anderson, and Lubomir Bourdev. Learned video compression. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3454–3463, 2019.

[36] Daniel L Ruderman. The statistics of natural images. *Network: computation in neural systems*, 5(4):517–548, 1994.

[37] Eero P Simoncelli and Bruno A Olshausen. Natural image statistics and neural representation. *Annual review of neuroscience*, 24(1):1193–1216, 2001.

[38] Hossein Talebi and Peyman Milanfar. Nima: Neural image assessment. *IEEE Transactions on Image Processing*, 27(8):3998–4011, 2018.

[39] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy image compression with compressive autoencoders. *arXiv preprint arXiv:1703.00395*, 2017.

[40] Pankaj Topiwala, Madhu Krishnan, and Wei Dai. Performance comparison of vvc, av1, and hevc on 8-bit and 10-bit content. In *Applications of Digital Image Processing XLI*, volume 10752, page 107520V. International Society for Optics and Photonics, 2018.

[41] Martin J Wainwright and Eero Simoncelli. Scale mixtures of gaussians and the statistics of natural images. *Advances in neural information processing systems*, 12:855–861, 1999.

[42] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003.

[43] Xiaoxia Wu, Ethan Dyer, and Behnam Neyshabur. When do curricula work? *arXiv preprint arXiv:2012.03107*, 2020.

[44] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. Video enhancement with task-oriented flow. *International Journal of Computer Vision*, 127(8):1106–1125, 2019.

[45] Ren Yang, Fabian Mentzer, Luc Van Gool, and Radu Timofte. Learning for video compression with hierarchical quality and recurrent enhancement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6628–6637, 2020.

[46] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[47] Fan Zhang, Angeliki V Katsenou, Mariana Afonso, Goce Dimitrov, and David R Bull. Comparing vvc, hevc and av1 using objective and subjective assessments. *arXiv preprint arXiv:2003.10282*, 2020.

[48] Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. Loss functions for image restoration with neural networks. *IEEE Transactions on computational imaging*, 3(1):47–57, 2016.