Escaping The Complexity-Bitrate-Quality Barriers Of Video Encoders Via Deep Perceptual Optimization

A. Chadha, R. Anam, I. Fadeev, V. Giotsas, and Y. Andreopoulos

iSIZE, London, U.K.

ABSTRACT

We extend the concept of learnable video precoding (rate-aware neural-network processing prior to encoding) to deep perceptual optimization (DPO). Our framework comprises a pixel-to-pixel convolutional neural network that is trained based on the virtualization of core encoding blocks (block transform, quantization, block-based prediction) and multiple loss functions representing rate, distortion and visual quality of the virtual encoder. We evaluate our proposal with AVC/H.264 and AV1 under per-clip rate-quality optimization. The results show that DPO offers, on average, 14.2% bitrate reduction over AVC/H.264 and 12.5% bitrate reduction over AV1. Our framework is shown to improve both distortion- and perception-oriented metrics in a consistent manner, exhibiting only 3% outliers, which correspond to content with peculiar characteristics. Thus, DPO is shown to offer complexity-bitrate-quality tradeoffs that go beyond what conventional video encoders can offer.

Keywords: visual quality, neural networks, video coding, AVC/H.264, AV1

1. INTRODUCTION

Advances in video encoding have traditionally leveraged on reducing the encoding bitrate for a given signal fidelity that has traditionally been expressed via average peak signal-to-noise ratio (PSNR). However, after questioning the validity of PSNR as a good indicator of visual quality across various types of content,[1] many groups have begun making intensive efforts towards the derivation of quality metrics that are better suited to human visual perception. These have now matured to metrics like the structural similarity index metric (SSIM) and its multiscale variant (MS-SSIM),[2] the video multimethod assessment fusion (VMAF),[2] and DeepQA methods.[3–5] These have been shown to be significantly more accurate in assessing visual quality and fidelity to the video source than PSNR and other such low-level signal distortion metrics. They also open up the opportunity for advanced perceptual optimization that goes beyond what is achievable with encoding recipe tuning and per-scene rate-quality optimization in video encoders.

In this paper, after a review of related work, we outline the key principles of a deep perceptual optimizer framework (DPO) that trains a "precoding" neural network to perform a pixel-to-pixel non-linear mapping prior to encoding with any standard or proprietary encoder (Section 3). In order to ensure we push the utilized encoders to their limits, our benchmarking with H.264/AVC and AV1 uses per-scene rate-quality optimization over multiple resolutions and bitrates[6] (Section 4). Our experiments show that DPO and H.264 or AV1 obtains significant rate savings over the same baseline encoder in terms of SSIM, VMAF and VMAF with its recently-introduced enhancement-limiting modifications. This allows for complexity-bitrate-quality tradeoffs that escape the barriers of video encoders, examples of which are given in Section 5. Finally, Section 6 concludes the paper.

Y. A. is Director at iSIZE and also Professor at University College London (UCL), U.K.; V. G. is Technical Consultant at iSIZE and also Lecturer at Lancaster University, U.K.; all proposals of this paper comprise intellectual property of iSIZE that is under patent-pending status; contact information for all authors: info@isize.co.

2. RELATED WORK

There has been significant work in learned image[7–10] and video[11–14] compression that strives to replace the entire coding pipeline with optimized autoencoders. For example, Lu *et al.*[11] replace entire blocks of a standard video codec with neural networks, such as replacing motion compensation with an optical flow warping network. Typically, rate is optimized in a latent space that is learned by a neural network-based encoder and not the frequency-domain transformed space of the image that is adopted by standard image or video codecs. However, all current results in learned video compression do not surpass standards like AVC or HEVC when the latter are utilized with their most advanced configurations.[15,16] In addition, more advanced encoders like AOMedia Video 1 (AV1) and AV2 encoders and early implementations of MPEG's Versatile Video Coding (VVC) standard already include neural components for optimized encoding tool selection,[17–19] while allowing for real-time decoding on CPU-based commodity devices like tablets and mobile phones. We therefore design our proposed DPO approach as a pixel-to-pixel precoding mechanism for any such standard hybrid encoder. This means our proposal does not alter the encoding or decoding sides in any way and, unlike recent proposals on content-adaptive resolution selection,[20,21] it does not even impose any changes to the video resolution.

Blau *et al.*[22] recently highlighted how optimization of image-restoration algorithms for distortion measures such as PSNR or SSIM is not sufficient for ensuring good perceptual quality of the results. There exists a perception-distortion plane that cannot be attained by any algorithm and, in the proximity of this plane, distortion must be traded off for perceptual quality or vice-versa. Contrary to recent video compression work[11–14] that only accounts for the rate-distortion tradeoff and hence validates on SSIM and PSNR, we are the first to propose the concept of joint optimization over perceptual quality, rate and distortion, with the intention of achieving a good operating point within the perception-distortion-rate-complexity space. We therefore validate our proposal on a range of quality metrics that span the perception-distortion space: SSIM, VMAF with its recently-introduced enhancement-limiting parameters* that are designed to penalize preprocessing methods that cause perceptual deviations from the source content†,[23] and the original VMAF metric,[24] which is mostly oriented towards perceptual quality rather than signal distortion.

3. DEEP PERCEPTUAL OPTIMIZER

In this section, we present our deep perceptual optimization for video precoding. Since the full technical details of our training process require an extensive exposition that goes beyond the scope of the present paper, we provide a summary presentation here and will present further details in subsequent publications. Essentially, the objective of our perceptual optimization framework is to provide a perceptually-enhanced and rate-controlled representation of the input frames via a learnable preprocessing or "precoding". The precoding must have some level of encoder-awareness so that it can incorporate the effects incurred from different codec settings during deployment, such as different modes for block based motion compensated prediction and quantization of error frame information. Therefore, our optimization framework must model (or "virtualize") the multiple processing components of a standard video coding pipeline – from the inter/intra prediction blocks, to the spatial transform and quantization. These coding blocks are appended to our video precoding, such that they operate in the precoding space on precoded frames. In this way, we can train the precoding *end-to-end* with our virtual encoder representation and effectively perform perceptual-rate-distortion optimization on the precoding parameters with stochastic gradient descent. We leverage on our proposed loss formulations to create a generalized model that approximates the rate-distortion behaviour of standard codecs. During 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 1. The training framework is in open loop configuration, which simplifies training, as there is no need for a decoded frames buffer containing previously-reconstructed frames as in the closed-loop case. 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 block represents the precoding network that maps input video frame $\boldsymbol{x}_t$ at time t to precoded frame $\boldsymbol{p}_t$. The orange block represents the components for inter (motion estimation and compensation) and intra prediction, which outputs

*see pull request #601 of Netflix VMAF repository available online at https://github.com/Netflix/vmaf

†see commit 615dc24579d531cb3a2c9627ab25a3026f9e2b47 of AOM repository https://aomedia.googlesource.com/aom/

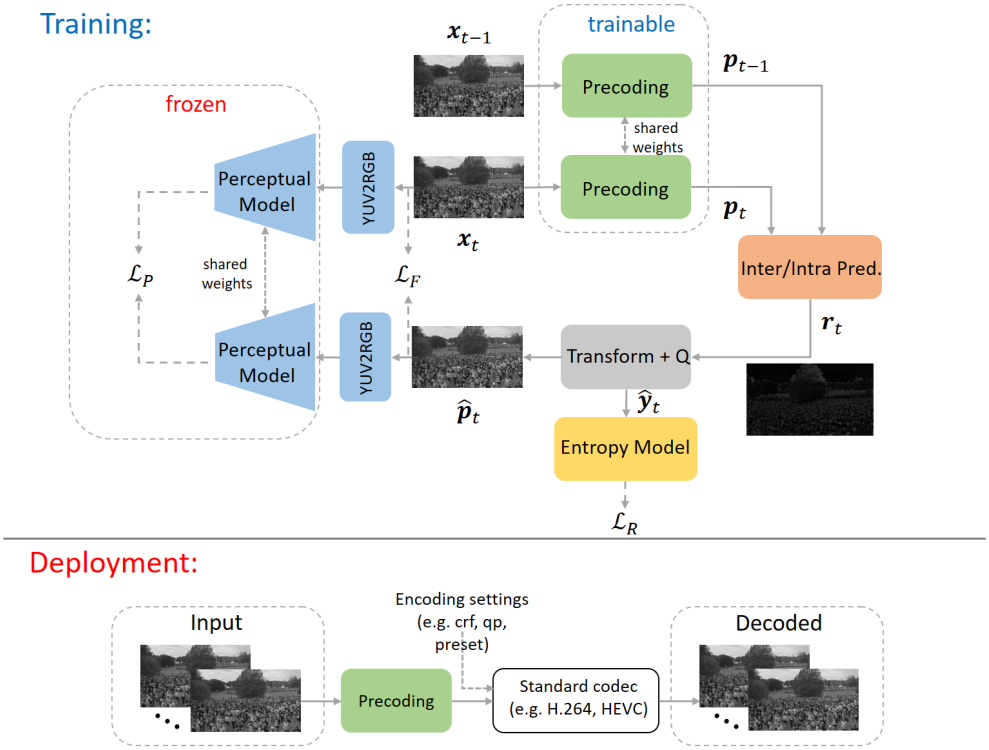


Figure 1: Deep perceptual optimizer framework for training perceptually-enhanced & rate-controlled representation of input frames via a learnable precoding. Dashed arrows represent optional components.

a predicted frame $\tilde{\boldsymbol{p}}_t$ and residual frame $\boldsymbol{r}_t$ by performing block matching between the current and reference precoded frames, $\boldsymbol{p}_t$ and $\boldsymbol{p}_{t-1}$ respectively. The grey block represents the spatial transform and quantization components for encoding and compressing the residual. The residual frame is transformed to the frequency domain output 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 precoding network to enhance the perceptual quality of reconstructed frame $\hat{\boldsymbol{p}}_t$. We next go through the design of each of these components.

3.1 Precoding

The input video frames are first processed individually by a precoding block, represented in green in Figure 1. The precoding block $F(\boldsymbol{x};\Theta)$ comprises a pixel-to-pixel mapping F, with associated parameters Θ. For efficient precoding and deployment, the input frame is luminance only. The luminance channel contains most of the image information and is the main contributor to perceptual sharpness and bandwidth, 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[25] 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.

3.2 Inter and Intra Prediction

The precoding 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 adaptively divides the frame into variable-sized macroblock partitions and sub-partitions, typically varying from 16×16 to 4×4. Similarly, in our virtual codec, the precoded frame $\boldsymbol{p}_t$ is first divided into a set of blocks of 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. We emulate the variable-sized partitioning schemes of standard encoders by randomly selecting $K \in \{4, 8, 16\}$ during training. 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:

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

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 (1) 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:

$$(m_1^*, m_2^*) = \arg\min_{(m_1,m_2)}(\epsilon(m_1, m_2)) \tag{2}$$

The displacement or motion vector $\boldsymbol{m}^* = (m_1^*, m_2^*)^{\mathrm{T}}$ is encoded for each block in the current frame and the arg min in (2) is converted into a differentiable form for backpropagation by approximating the one-hot matrix, representing the optimal block location, with a continuous categorical distribution over all blocks in the local search space (via the softmax function). The predicted frame $\tilde{\boldsymbol{p}}_t^{\text{inter}}$ is then configured as:

$$\tilde{\boldsymbol{p}}_t^{\text{inter}}(n_1+k_1, n_2+k_2) = \sum_{(m_1,m_2)} \mathbf{1}_{(\boldsymbol{m}^*)}(m_1, m_2).\boldsymbol{p}_{t-1}(n_1+k_1+m_1, n_2+k_2+m_2) \tag{3}$$

where $\mathbf{1}_{(\boldsymbol{m}^*)}$ is an one-hot matrix with the unity value set at the position $\boldsymbol{m}^*$. The residual frame $\boldsymbol{r}_t$ is simply equal to the difference between the predicted frame and current frame: $\boldsymbol{r}_t = \boldsymbol{p}_t - \tilde{\boldsymbol{p}}_t^{\text{inter}}$. For intra prediction, we follow a similar approach for generating $\tilde{\boldsymbol{p}}_t^{\text{intra}}$, 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). In this way, we are able to emulate all forms of intra prediction modes.

3.3 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,[26] 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. Specifically, any value of Q_{step} can be derived from the first 6 values of QP as $Q_{\text{step}}(\text{QP}) = 2Q_{\text{step}}(\mod(\text{QP}, 6)).\text{floor}\left(\frac{\text{QP}}{6}\right)$. We denote the quantized frame as $\hat{\boldsymbol{y}}_t$ and approximate the non-differentiable rounding operation via additive uniform noise during training. 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.[27] 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 $\tilde{\boldsymbol{p}}_t + \hat{\boldsymbol{r}}_t$.

3.4 Entropy Model

Given that we aim to optimize our precoding on rate, we must minimize the number of bits required to encode the DCT transformed and quantized frame $\hat{\boldsymbol{y}}_t$. In order to compute the number of bits in a differentiable manner (amenable to backpropagation), we approximate the actual rate with the entropy computed on the DCT subbands. We model each subband as an independent probability density function and assume independence between subbands. We follow Balle *et al.*[8] and approximate the entropy computation with univariate non-parametric density models $R_s(.;\Phi)$, where s denotes the subband. Approximating the entropy in this manner means that: *(i)* we are assuming a factorized representation; *(ii)* we are not accounting for context-adaptive entropy coding, which is now a standard tool of advanced video encoding. Nonetheless, we find empirically that we are still able to minimize the bitrate with the precoding network trained under this approximation.

3.5 Perceptual Model

We aim to optimize our precoding for perceptual enhancement of the decoded input frame representations $\hat{\boldsymbol{p}}_t$. To this end, we follow Talebi *et al.*[28] and first pre-train a no-reference image quality assessment (IQA) model $P(.;\Psi)$ for predicting the distribution over the ACR scale, which maps human image ratings from poor (1) to excellent (5). Specifically, we fine-tune a VGG-16 model that is pretrained on ImageNet.[29] The fully connected layers are removed and replaced with global average pooling and single fully connected layer with 5 neurons, each representing an image rating, and softmax function to map the output to a distribution. Once trained, the IQA model is then frozen for the precoding optimization. We note that given that our perceptual model is trained on human-rated RGB images, it is necessary in our perceptual optimization framework to first convert the luminance frame $\hat{\boldsymbol{p}}_t$ to RGB frame $\hat{\boldsymbol{p}}_t^{\text{RGB}}$. 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, $\boldsymbol{x}_t^{\text{RGB}}$.

3.6 Loss Functions

Our overall objective is to train our precoding $F(\boldsymbol{x}_t;\Theta)$ to perform perceptual-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, perceptual quality and distortion of the decoded frames during deployment with a standard video codec. To this end, we train the precoder end-to-end with the building blocks of our optimization framework and a perceptual loss ($\mathcal{L}_P$), rate loss ($\mathcal{L}_R$) and fidelity loss ($\mathcal{L}_F$). The overall loss function for training the precoder can thus be written as a weighted summation: $\mathcal{L}(\boldsymbol{x}_t, \hat{\boldsymbol{p}}_t; \Theta) = \gamma\mathcal{L}_P + \lambda\mathcal{L}_R + \mathcal{L}_F$, where γ and λ are the perceptual and rate coefficients respectively. By jointly training all three loss components, we find the operating point close to the rate-distortion convex hull whilst maintaining or enhancing the perceptual quality relative to the input frame $\boldsymbol{x}_t$. Our fidelity loss comprises the combination of multiscale SSIM (as defined by Wang *et al.*[30]) and the L_1 distance between $\boldsymbol{x}_t$ and $\hat{\boldsymbol{p}}_t$, which is good for preserving luminance. The rate loss is summed over all DCT subbands; per-subband, it comprises the log-likelihood of values of $\boldsymbol{y}_t$. Finally, our perceptual loss expresses the need for the perceptual quality of $\hat{\boldsymbol{p}}_t^{\text{RGB}}$ to be greater than $\boldsymbol{x}_t^{\text{RGB}}$. We can quantify perceptual quality with our pre-trained perceptual model P, by computing the mean opinion scores (MOS) as the expectation over the output ACR distributions. Our perceptual loss thus optimizes the MOS score of our precoded frame representation $P(\hat{\boldsymbol{p}}_t^{\text{RGB}})$ over the input frame $P(\boldsymbol{x}_t^{\text{RGB}})$.

4. EXPERIMENTAL RESULTS

4.1 Implementation Details

The perceptual model P is first trained on a proprietary no-reference IQA dataset using stochastic gradient descent with momentum set to 0.9 and an initial learning rate of 1×10^{-3}. Our IQA dataset comprises a large number of training images with authentic distortions captured 'in the wild' and associated human opinion scores. The perceptual model is then frozen and the deep precoding is trained in an end-to-end manner as per the design of Figure 1 and loss functions of Section 3.6. Let us denote $\text{Conv}(f, c, r)$ as convolutional layers, with f being the kernel size, c the number of channels and r the dilation rate. The precoding architecture can thus be expressed as: $\text{Conv}(3,16,1) \rightarrow \text{Conv}(3,16,1) \rightarrow \text{Conv}(3,16,2) \rightarrow \text{Conv}(3,16,4) \rightarrow \text{Conv}(3,16,8) \rightarrow \text{Conv}(3,1,1)$. Each convolutional layer is followed by a parametric ReLu activation function. Inspired by human foveal vision and

focus-of-attention, we only pass fixed size crops of size 512×512, rather than the full frame, for efficient training. 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 precoding is trained with Adam optimizer and initial learning rate of 1×10^{-4} for 40k steps, with the learning rate decayed by a factor of 10 after 20k steps. We derive our precoder model variants, by adjusting the rate coefficient $\lambda \in [0.001, 0.1]$ and perceptual coefficient $\gamma \in [0.1, 1]$, which controls the perception-distortion-rate tradeoff. While numerous models can be generated, we only utilize two models, which we term enh0m and enh3m; they correspond to setting $\gamma = 0.1$ in both cases and $\lambda = 0.01$ for enh0m and $\lambda = 0.001$ for enh3m. At deployment, we only retain the precoding that comprises the learned pixel-to-pixel mapping of these two models, i.e., the virtual codec is replaced with a standard video codec.

In terms of runtime complexity, for each of the two utilized models our current implementation in 16-bit floating-point arithmetic achieves 8.4ms per frame for 1080p input on an NVIDIA Tesla T4 GPU, and can easily run in real time for both full HD and ultra HD, with the latter requiring two GPUs. For CPU implementation, quantizing the inference model via Intel's OpenVINO framework (with the 'AccuracyAware' quantization method and 'accuracy' preset) and optimizing load times and padding at the borders, yields up to 456% speedup over their floating-point counterparts: on an Intel Xeon platinum 8259cl CPU, we achieve up to 11.4 fps for 8-bit quantized precoding networks versus 2.5 fps for 32-bit floating point. We can therefore achieve real-time performance for 1080p video with a cluster of 5 or 6 such CPUs. The impact of 8-bit quantization on accuracy of our precoding can be limited, and on-going work will assess the deployment efficiency of such designs on Intel CPUs.

4.2 Experimental Setup for Bjontegaard Delta-Rate Results

We present a detailed evaluation of different models against state-of-the-art implementations of two encoder generations used in adaptive streaming systems: the Advanced Video Coding (AVC) and AOMedia Video 1 (AV1), utilizing the libx264 and aomenc open implementations of these standards. This makes our results cross-comparable under realistic encoder designs that are deployed widely within adaptive streaming systems.

We report on tests with XIPH and CDVL sequences[‡] (representing prime content) and YouTube UGC 1080p 'LiveMusic' and 'Sports' (as examples of user-generated content[31]) using the x264 H.264/AVC recipe:

```
ffmpeg -y -i XIPH_video_in.y4m -vf scale=WxH:flags=lanczos -c:v libx264 -profile:v high
-threads 4 -preset Ph264 -crf C -refs 5 -g 150 -tune ssim -x264opts ssim=1 -keyint_min 150
-sc_threshold 0 -f mp4 XIPH_video_out.mp4
```

and aomenc AV1 recipe (Lanczos downscaling takes place losslessly prior to encoding with aomenc):

```
aomenc --passes=2 --pass=1 --fpf=aom1.log --target-bitrate=B --cpu-used=Pav1 --threads=8
--tile-columns=1 --tile-rows=0 --kf-max-dist=150 -o "video_out.webm" "video_in.y4m" 2>1
```

```
aomenc --passes=2 --pass=2 --fpf=aom1.log --target-bitrate=B --cpu-used=Pav1 --threads=8
--tile-columns=1 --tile-rows=0 --kf-max-dist=150 -o "video_out.webm" "video_in.y4m" 2>1
```

The utilized CRF values for H.264 encoding are: `C`$\in\{$`18, 22, 26, 30, 34, 38, 42`$\}$. For AV1, the utilized bitrates are: `B`$\in\{$`138, 230, 385, 642, 1070, 1800, 3000, 5000, 8000`$\}$ kbps, i.e., increase by 40% per step. In terms of complexity presets, we are testing with `Ph264`$\in\{$`slow,veryslow`$\}$ and `Pav1`$\in\{$`3,5`$\}$. All utilized resolutions `WxH` are produced by FFmpeg Lanczos used on the source video or on the DPO output. They are: 1080p, 720p, 540p, 432p, 360p, 288p, 216, 144p. All decoded results are upscaled with FFmpeg bicubic to 1080p prior to quality measurement via libvmaf. All Bjontegaard delta-rates (BD-rates)[32] 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 standard BD-rate measurements to find average rate saving per sequence (positive BD-rates indicate rate loss). To avoid skewing the BD-rate averages from the use of very-low or very-high bitrate points, we keep results within the following ranges: • $40 \leq$ VMAF ≤ 96; • $88 \leq$ SSIM ≤ 99, which correspond to commercially-viable quality ranges and avoid excessively-high bitrates that would not be used in practice. We have also confirmed that, under these limits, the calculated BD-rates for the different quality metrics correspond

[‡]XIPH source material from https://media.xiph.org/video/derf/ and CDVL material from https://www.cdvl.org/

to similar bitrate ranges and can thus be averaged together. This type of measurement corresponds to a dynamic-optimizer type of BD-rate measurement per clip[6] for the quality regimes usable in video delivery. That is, for each point in the bitrate-quality range, our method for evaluation finds the optimum resolution and CRF (or bitrate) of that resolution to use per clip. For each case reported, the DPO+codec results correspond to the same resolutions and encoding recipe used as for the codec results. For all experimental points per sequence, only a single-pass is required per frame for each of our enh0m and enh3m models, and the latency is only 1 frame. VMAF and SSIM were computed using the Netflix libvmaf library. With regards to VMAF, we used the standard 0.6.1 library with and without the latest update that includes parameters: `ADM_ENHN_GAIN_LIMIT` and `VIF_ENHN_GAIN_LIMIT`, which are designed to eliminate the gain offered by nave preprocessing.[2,3] When setting these parameters to 1.0 in the libvmaf code, the results are reported as Anti-Hacking VMAF (AH-VMAF). For reference we also report the conventional VMAF score without these limits (shown as VMAF).

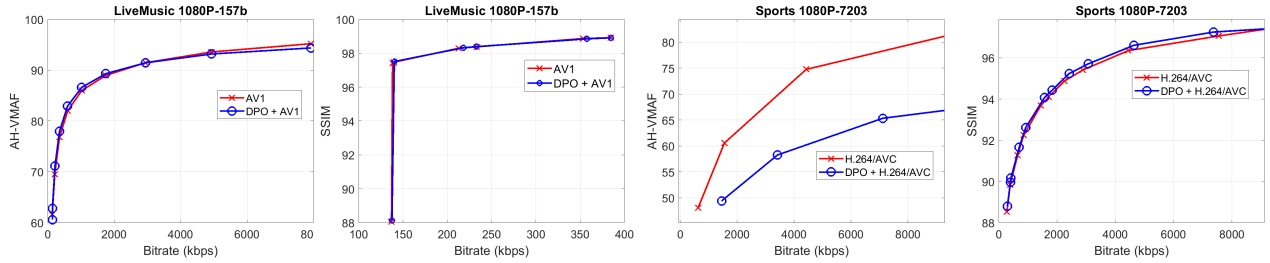
4.3 BD-Rate Results with H.264/AVC and AV1

The results of Table 1-Table 4 show that the average rate saving over VMAF, AH-VMAF and SSIM for both H.264 and AV1 standards is over 10%. As expected, our gains are higher on metrics that are increasingly perception-oriented rather than distortion-oriented: on VMAF, our framework offers 23% to 30% saving; on AH-VMAF, they are between 6% to 12% and on SSIM they are 1.3%-3.6% for AV1 and 3.4%-4.7% for H.264. 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 perceptual dimensions (VMAF), as well as on AH-VMAF that strikes a balance between the two. In summary: *(i)* these savings are achieved on top of the dynamic optimizer approach,[6] which already pushes the rate-quality curves of each baseline encoder to its performance limits; *(ii)* our framework demonstrates consistency across multiple encoders, encoding presets and multiple metrics with very few outliers.

To contrast these results with hand-crafted alternatives for perceptual enhancement that are based on sharpening filters, we also run the same comparisons as for Table 1-Table 4 by replacing our DPO with: *(i)* the FFmpeg recipe `-vf eq=brightness=0.0:contrast=1.075,unsharp=5:5:1.05:5:5:0.0` prior to x264 encoding at each resolution and CRF, which was proposed as a mild contrast adjustment and sharpening alternative in a recent article of J. Ozer (Streaming Media Mag., March 2020); *(ii)* the `-tune=vmaf_with_preprocessing` for VMAF enhancement within aomenc, as was proposed recently by Google. Table 5 shows the results of these sharpening based approaches. Evidently, such sharpening recipes are very well suited for the perceptual side (bitrate reductions for VMAF range between 32% to 34%), but perform poorly on the distortion side (SSIM and AH-VMAF drop significantly, with the exception of aomenc cpu=3 on XIPH & CDVL, where bitrate reduction of 19% is attained for AH-VMAF). This leads to their BD-rate averages on all metrics (bottom of Table 5) being negligible or even detrimental in terms of bitrate. Therefore, unlike our proposal, such sharpening approaches are found to be very biased towards the perceptual front. This may make their deployment more challenging than the proposed DPO that shows significantly more consistent behavior across all quality metrics.

Analysis of Outliers: The XIPH and CDVL sequences exhibited no outlier behaviors. However, a small number of UGC sequences exhibited abnormal encoding behaviors and were excluded from our averages in order not to skew our results. These correspond to 6 sequences out of the 93 sequences used. Out of the 6 videos, 'LiveMusic_1080P-157b', 'LiveMusic_1080P-59b3' and 'LiveMusic_1080P-3e1a' exhibited gains for DPO preprocessing, but had sharp jumps in SSIM and/or VMAF for the AV1 encoder, as seen in the example of Figure 2(a-b). This makes their BD-rate calculations unreliable for AV1. Hence, these sequences were excluded from the averages of Table 4. The remaining 3 sequences: 'LiveMusic_1080P-7948', 'Sports_1080P-43e2' and 'Sports_1080P-7203' (so around 3% of our tests), exhibited significant discrepancies between the VMAF/AH-VMAF measurements and the SSIM measurements for both encoders: one set of metrics was increasing in comparison to the codec but the other was decreasing [Figure 2(c-d)], or all metrics were saturating. Detailed inspection revealed that these sequences exhibit one or more of the following issues: extreme temporal flicker, significant spatial aliasing, intensity saturation, large number of frames that comprise black background with letters. Such peculiarities can be handled with a pass-through (or 'null') model that will detect that metrics are not improving in a coherent manner and will just pass the original source frames through to the output of the DPO without any processing. If such detection of metrics' abnormalities is implemented, it also raises the interesting prospect of using our framework as a neural 'visual inspector', which can detect spatio-temporal

abnormalities in the input video for the benefit of the content service provider (e.g., recommendations on visual aesthetics of the source content can be made prior to compression). Since we did not include such a detection and pass-through option in our results, these 3 sequences are excluded from our averages. Finally, it is interesting to note that the VMAF and AH-VMAF exhibited significantly more robust behaviour than SSIM. This illustrates their advantage and robustness in comparison to SSIM (or PSNR).



(a) aomenc cpu=3 (b) aomenc cpu=3 (c) x264 veryslow (d) x264 veryslow

Figure 2: Parts (a) and (b) show an example of normal behaviour of DPO; however, the sharp jump in SSIM leads to erroneous BD-rate calculations, and is therefore excluded from our averages. Parts (c) and (d) show an example of abnormal behaviour of DPO, where SSIM follows the expected trend but AH-VMAF is saturated; this corresponds to a case that can be dealt with via pass-through DPO.

5. COMPLEXITY-BITRATE-QUALITY TRADE-OFFS

Since our approach offers more than 11% BD-rate reduction for the combination of VMAF, AH-VMAF and SSIM, it is interesting to explore its effects across multiple encoders and encoding settings. Table 6 and Table 7 show average BD-rates over VMAF, AH-VMAF and SSIM when transitioning between encoders and encoding recipes with and without DPO. The boldface & underlined numbers show the BD-rate when transitioning from a baseline encoding setting to another encoding setting (target) that is the immediately less-complex; the corresponding speed ratios between 'Baseline' and 'Target' encoders are given in Table 8, as measured in multi-CPU cloud instances[§].

For example, inspection of Table 6 and Table 7 shows that downgrading from x264 veryslow to x264 slow incurs 7.75% increase in bitrate. Downgrading aomenc cpu=5 to x264 veryslow comes at 43.63% bitrate increase, showcasing the substantial benefits offered by aomenc in comparison to x264. However, these transitions come at 2-fold and 14.5-fold increase in encoding speed, as shown in Table 8. Hence, if this loss in performance can be ameliorated, this may make the benefit of speed improvements overcome the bitrate detriment, especially when considering a number of scenarios involving high-volume video encoding. The use of DPO makes such amelioration possible, as shown in Table 7. For example, switching from x264 veryslow to DPO + x264 slow preset allows for this two-fold increase in encoding speed while simultaneously providing for 7.89% *reduction* in bitrate (cell 2,1 of Table 7). Similarly, switching from aomenc cpu=3 to DPO + aomenc cpu=5 allows for 2.49% bitrate saving and 1.88-fold speedup (cell 4,3 of Table 7 and Table 8). Taking this approach to cross-encoder switching, going from aomenc cpu=5 to DPO + x264 veryslow incurs 24.46% overhead and allows for the aforementioned 14.5-fold speedup in encoding. While the 24.46% loss of encoding efficiency is still substantial, it is significantly less than the aforementioned 43.63% difference between aomenc cpu=5 and x264 veryslow.

Such transitions can also operate in reverse. For example, switching from x264 'slow' to DPO + aomenc cpu=3 has the daunting prospect of slowing down encoding speed to only 2% of that of x264. However, it can also allow for 47.5% saving in bitrate, which may make such a transition much more worthwhile to attempt for highly-popular video content. Similarly, switching from x264 'slow' to DPO + x264 'veryslow' comes at

[§]Average runtimes are taken over multiple runs on an AWS g4dn.8xlarge instance (32 vCPUs, 128GB RAM) with 8-thread execution of the x264 and aomenc encoder versions available from their respective repositories at the time of this writing. While speedup ratios between different encoding recipes and encoders will evolve over time based on the degree of optimization devoted to each implementation, both encoders are now reasonably mature and stable, and they are used widely as open benchmarks for speed and encoding efficiency.

Table 1: Bjontegaard delta-rate results for DPO enh0m+enh3m models and H.264/AVC in terms of metrics that are increasingly perceptual in nature: SSIM, AH-VMAF (`ADM_ENHN_GAIN_LIMIT = VIF_ENHN_GAIN_LIMIT = 1.0`) and VMAF. The first 16 sequences are from the XIPH website; the remaining sequences are from CDVL.

x264 Encoder Preset	slow			veryslow		
Sequence	SSIM (%)	AH-VMAF (%)	VMAF (%)	SSIM (%)	AH-VMAF (%)	VMAF (%)
rush_field_cuts_1080p30	-3.74	-10.50	-21.81	-3.16	-9.20	-21.82
rush_hour_1080p25	-0.98	-12.61	-25.72	-2.04	-12.60	-25.43
sunflower_1080p25	-7.68	-13.4	-24.91	-6.82	-12.16	-21.62
tractor_1080p25	-6.73	-14.29	-21.97	-7.05	-13.73	-21.85
touchdown_pass_1080p30	-7.23	-13.46	-26.77	-5.44	-13.12	-26.39
riverbed_1080p25	-1.78	-7.04	-13.76	-1.73	-7.17	-14.11
west_wind_easy_1080p30	-4.42	-10.28	-21.19	-2.64	-9.81	-20.54
aspen_1080p30	-5.26	-9.10	-25.94	-4.97	-10.86	-26.05
blue_sky_1080p25	-4.97	-8.33	-17.57	-4.40	-8.23	-17.35
controlled_burn_1080p30	-6.26	-12.12	-30.92	-4.38	-10.85	-31.65
crowd_run_1080p50	-5.01	-10.39	-19.45	-5.26	-10.28	-19.15
ducks_take_off_1080p50	-9.39	-17.46	-27.32	-8.31	-18.06	-28.75
old_town_cross_1080p50	-6.07	-14.11	-31.84	-4.52	-13.92	-31.29
park_joy_1080p50	-8.18	-12.66	-22.72	-7.03	-12.42	-22.34
pedestrian_area_1080p25	-4.38	-14.38	-22.66	-4.77	-13.51	-21.95
red_kayak_1080p30	-1.42	-10.92	-18.86	-1.46	-10.54	-18.56
vqeghd4_csrc11_original	-1.25	-11.35	-23.82	-1.18	-9.97	-22.36
vqeghd4_csrc12_original	-6.49	-13.91	-31.12	-3.65	-13.76	-29.79
vqeghd4_csrc13_original	-5.84	-14.17	-25.43	-5.78	-14.65	-25.16
vqeghd4_csrc14_original	-5.15	-8.73	-24.13	-4.26	-8.69	-23.94
vqeghd4_src01_original	-1.79	-12.28	-26.41	-3.39	-11.99	-25.94
vqeghd4_src02_original	-3.53	-9.94	-20.45	-3.31	-9.54	-19.94
vqeghd4_src03_original	-5.21	-14.55	-32.32	-4.88	-14.3	-33.18
vqeghd4_src04_original	-1.69	-7.92	-19.34	-1.18	-7.33	-20.25
vqeghd4_src05_original	-6.32	-17.18	-34.25	-6.51	-19.57	-40.62
vqeghd4_src06_original	-4.06	-12.27	-29.18	-4.52	-16.16	-30.22
vqeghd4_src07_original	-3.99	-12.18	-21.12	-4.19	-11.98	-20.83
vqeghd4_src08_original	-3.46	-16.42	-28.55	-2.79	-15.21	-28.05
vqeghd4_src09_original	-4.00	-15.78	-30.97	-4.06	-14.23	-35.33
vqeghd5_csrc11_original	-1.25	-11.34	-23.81	-1.18	-9.97	-22.36
vqeghd5_csrc12_original	-6.49	-13.91	-31.12	-3.66	-13.76	-29.79
vqeghd5_csrc13_original	-5.84	-14.17	-25.43	-5.78	-14.62	-25.14
vqeghd5_csrc14_original	-5.15	-8.73	-24.13	-4.26	-8.69	-23.94
vqeghd5_src01_original	-2.99	-12.22	-25.03	-4.00	-11.45	-24.93
vqeghd5_src02_original	-3.15	-5.67	-28.57	0.01	-0.77	-30.52
vqeghd5_src04_original	-5.07	-13.38	-21.24	-3.77	-13.38	-21.16
vqeghd5_src05_original	-9.35	-12.90	-28.43	-5.25	-10.54	-26.78
vqeghd5_src06_original	-2.23	-9.21	-14.93	-1.64	-9.51	-14.83
vqeghd5_src08_original	-5.21	-19.37	-35.85	-4.15	-18.62	-35.74
vqeghd5_src09_original	-3.81	-12.16	-24.33	-5.05	-12.58	-24.61
Average	**-4.67**	**-12.27**	**-25.08**	**-4.06**	**-11.94**	**-25.11**
Av. AH-VMAF & VMAF	–	**-18.68**		–	**-18.53**	
Average of all three	**-14.00**			**-13.70**		

Table 2: BD-rate results for DPO enh0m+enh3m models and H.264/AVC. The sequences are from YouTube UGC dataset.[31] Videos marked with an asterisk (*) are excluded from the averages and are discussed separately.

x264 Encoder Preset	slow			veryslow		
Sequence	SSIM (%)	AH-VMAF (%)	VMAF (%)	SSIM (%)	AH-VMAF (%)	VMAF (%)
LiveMusic_1080P-14af	-0.90	-9.93	-20.30	-1.65	-9.74	-19.86
LiveMusic_1080P-157b	-5.63	-13.40	-28.51	-6.03	-12.98	-27.77
LiveMusic_1080P-1ace	-1.83	-8.55	-22.54	-3.15	-7.96	-22.54
LiveMusic_1080P-21dd	-3.52	-10.48	-23.63	-2.48	-10.25	-24.02
LiveMusic_1080P-28fe	-4.91	-12.68	-22.73	-3.02	-12.21	-23.17
LiveMusic_1080P-2930	-4.69	-11.07	-19.28	-4.15	-11.37	-18.74
LiveMusic_1080P-2b7a	-2.30	-10.15	-22.67	-6.09	-9.94	-22.61
LiveMusic_1080P-2f7f	-1.44	-10.26	-23.74	-1.24	-11.02	-21.76
LiveMusic_1080P-3549	-8.09	-10.34	-23.18	-8.16	-9.89	-21.80
LiveMusic_1080P-3e1a	-2.75	-10.29	-30.44	-1.98	-7.77	-28.47
LiveMusic_1080P-3f95	-7.11	-9.80	-21.68	-2.97	-5.70	-21.94
LiveMusic_1080P-48d5	-5.07	-11.58	-21.02	-5.47	-10.04	-17.79
LiveMusic_1080P-514e	-0.73	-7.71	-13.64	-1.84	-8.80	-13.44
LiveMusic_1080P-51f6	-2.74	-14.67	-23.17	-3.26	-12.53	-21.10
LiveMusic_1080P-541f	-5.17	-11.89	-33.50	-2.06	-11.52	-33.75
LiveMusic_1080P-59b3	-0.96	-7.04	-21.52	-0.42	-10.42	-23.70
LiveMusic_1080P-6b1c	-3.81	-6.99	-21.56	-3.41	-8.05	-22.10
LiveMusic_1080P-6bbe	-4.37	-7.75	-13.60	-4.17	-7.83	-14.93
LiveMusic_1080P-6d1a	2.61	-15.19	-33.22	0.97	-13.34	-33.59
LiveMusic_1080P-6fe2	-1.55	-7.29	-20.09	-5.30	-7.26	-19.98
LiveMusic_1080P-77e8	-0.95	-13.01	-52.66	3.73	-7.83	-49.50
LiveMusic_1080P-7948*	75.00	33.65	43.46	75.00	37.06	47.87
LiveMusic_1080P-7ead	-1.30	-15.97	-23.57	0.09	-11.90	-22.18
Sports_1080P-0063	-8.17	-21.18	-38.74	-9.01	-20.17	-41.67
Sports_1080P-0640	-0.99	-11.40	-23.27	-2.60	-10.64	-23.02
Sports_1080P-08e1	-4.24	-4.51	-17.73	-3.99	-4.33	-18.14
Sports_1080P-0d0c	-4.27	-10.73	-23.96	-4.76	-10.34	-22.98
Sports_1080P-15d1	-9.52	-16.05	-27.04	-8.00	-15.70	-27.31
Sports_1080P-19d8	-7.20	-9.69	-31.83	-4.74	-14.46	-32.72
Sports_1080P-1ae3	-5.01	-14.94	-31.92	-4.59	-14.27	-32.63
Sports_1080P-1bf7	-6.93	-16.40	-29.80	-6.59	-15.20	-27.90
Sports_1080P-1d78	-4.78	-21.92	-36.43	-2.63	-18.54	-36.59
Sports_1080P-241e	-2.50	-7.50	-22.70	-1.71	-6.78	-23.98
Sports_1080P-2524	-5.79	-21.60	-46.66	-4.19	-21.59	-47.17
Sports_1080P-28a6	-3.41	-10.49	-30.44	-2.38	-8.92	-29.07
Sports_1080P-2a21	-2.32	-9.25	-22.85	-1.55	-9.36	-22.49
Sports_1080P-3a3b	-4.36	-11.86	-25.35	-4.18	-11.96	-24.69
Sports_1080P-3db7	-4.48	-12.36	-32.14	-4.01	-10.79	-30.46
Sports_1080P-3eb0	-3.29	-11.94	-31.72	-4.84	-11.69	-31.53
Sports_1080P-43e2*	75.00	-4.09	-35.49	75.00	-7.62	-35.28
Sports_1080P-46ed	-6.22	-13.25	-32.56	-6.17	-12.65	-32.59
Sports_1080P-47e9	-6.31	-6.18	-30.42	-4.67	-4.77	-32.63
Sports_1080P-4978	-5.95	-18.55	-37.68	-4.54	-13.05	-35.72
Sports_1080P-49c5	-1.86	-9.86	-22.40	-2.57	-10.13	-22.10
Sports_1080P-4e05	-1.57	-15.50	-32.48	-3.37	-14.42	-31.67
Sports_1080P-53a0	-2.84	-19.48	-36.93	-3.32	-15.18	-36.61
Sports_1080P-5d25	-4.14	-20.74	-34.35	-3.59	-21.18	-35.13
Sports_1080P-6571	-4.75	-15.24	-32.43	-3.48	-14.45	-33.06
Sports_1080P-6710	-4.54	-10.06	-37.13	-4.84	-13.54	-38.68
Sports_1080P-679d	-3.42	-10.29	-31.52	-3.60	-10.12	-30.99
Sports_1080P-7203*	-4.50	75.00	75.00	-4.75	75.00	75.00
Sports_1080P-7584	-0.87	-10.40	-21.28	-1.09	-11.09	-21.28
Sports_1080P-76a2	3.99	-20.37	-40.86	7.12	-20.73	-44.43
Sports_1080P-78fa	-1.76	-12.10	-29.65	-2.18	-11.36	-28.88
Sports_1080P-7b51	-5.47	-14.19	-25.23	-4.77	-13.29	-26.14
Sports_1080P-7dba	-9.30	-7.84	-30.31	-9.52	-10.19	-33.46
Average	**-3.76**	**-12.30**	**-28.00**	**-3.44**	**-11.68**	**-27.93**
Av. AH-VMAF & VMAF	–	**-20.15**		–	**-19.81**	
Average of all three	**-14.69**			**-14.35**		

Table 3: BD-rate results for DPO enh0m+enh3m models and AV1. The first 16 sequences are from the XIPH website and the remaining sequences are from CDVL.

aomenc cpu setting	cpu=5			cpu=3		
Sequence	SSIM (%)	AH-VMAF (%)	VMAF (%)	SSIM (%)	AH-VMAF (%)	VMAF (%)
rush_field_cuts_1080p30	-4.07	-10.10	-21.62	-1.35	-6.05	-18.63
rush_hour_1080p25	-1.38	-11.97	-28.17	-2.55	-10.99	-29.29
sunflower_1080p25	-0.99	-8.30	-26.58	2.59	-3.70	-15.29
tractor_1080p25	-4.92	-13.12	-25.39	-4.49	-10.24	-20.67
touchdown_pass_1080p30	-4.61	-11.52	-22.37	-3.71	-7.60	-19.56
riverbed_1080p25	-3.53	-6.33	-14.19	-3.42	-4.45	-13.15
west_wind_easy_1080p30	-1.30	-4.78	-29.84	1.84	2.48	-31.29
aspen_1080p30	-0.14	-7.29	-28.69	-0.45	-7.11	-25.54
blue_sky_1080p25	-4.30	-6.98	-15.09	-2.49	-3.28	-11.55
controlled_burn_1080p30	-4.29	-6.80	-29.80	-2.14	-7.50	-31.07
crowd_run_1080p50	-3.46	-8.99	-16.91	-2.09	-7.35	-14.47
ducks_take_off_1080p50	-6.96	-16.68	-27.62	-4.38	-8.95	-23.61
old_town_cross_1080p50	-0.48	-6.47	-23.93	-2.63	0.45	-19.51
park_joy_1080p50	-6.90	-12.11	-22.99	-4.81	-8.42	-20.41
pedestrian_area_1080p25	-1.59	-12.78	-22.79	-3.52	-8.18	-20.69
red_kayak_1080p30	-3.74	-8.12	-20.91	-3.35	-5.81	-19.03
vqeghd4_csrc11_original	-3.14	-7.77	-22.20	-1.80	-5.71	-20.79
vqeghd4_csrc12_original	-6.33	-2.49	-30.32	4.29	-0.87	-29.13
vqeghd4_csrc13_original	-6.19	-12.39	-26.82	-3.36	-8.61	-26.17
vqeghd4_csrc14_original	-4.18	-7.12	-23.84	-3.81	-3.88	-20.63
vqeghd4_src01_original	-3.31	-12.59	-25.87	-1.30	-9.86	-25.27
vqeghd4_src02_original	-1.67	-7.33	-21.71	-2.97	-4.88	-19.56
vqeghd4_src03_original	-4.29	-9.01	-33.27	-2.71	-7.09	-32.41
vqeghd4_src04_original	-2.87	-12.63	-23.36	-1.32	-5.46	-16.18
vqeghd4_src05_original	-1.14	-16.06	-45.65	1.82	-14.15	-42.65
vqeghd4_src06_original	-6.82	-12.81	-35.96	-8.34	-12.93	-25.51
vqeghd4_src07_original	-6.52	-16.45	-27.09	-4.60	-7.47	-18.94
vqeghd4_src08_original	-3.02	-18.59	-31.85	1.54	-12.66	-28.57
vqeghd4_src09_original	5.55	-7.54	-27.52	10.05	0.49	-22.81
vqeghd5_csrc11_original	-3.14	-7.77	-22.20	-1.80	-5.71	-20.79
vqeghd5_csrc12_original	-6.33	-2.49	-30.32	4.29	-0.87	-29.13
vqeghd5_csrc13_original	-6.19	-12.39	-26.82	-3.36	-8.61	-26.17
vqeghd5_csrc14_original	-4.18	-7.12	-23.84	-3.82	-3.88	-20.63
vqeghd5_src01_original	-3.30	-10.33	-20.71	-3.13	-8.50	-18.93
vqeghd5_src02_original	-9.02	-0.07	-28.65	-0.04	6.56	-33.32
vqeghd5_src04_original	-3.46	-13.79	-21.19	-1.32	-11.39	-20.05
vqeghd5_src05_original	0.09	-0.19	-25.53	-0.98	3.21	-18.24
vqeghd5_src06_original	-5.31	-11.13	-16.99	-2.62	-8.57	-13.87
vqeghd5_src08_original	-4.87	-10.64	-43.43	-2.27	-6.53	-30.45
vqeghd5_src09_original	-2.60	-9.72	-23.85	-3.51	-9.17	-21.24
Average	**-3.62**	**-9.52**	**-25.90**	**-1.70**	**-6.08**	**-22.88**
Av. AH-VMAF & VMAF	–	**-17.71**		–	**-14.48**	
Average of all three	**-13.01**			**-10.22**		

Table 4: BD-rate results for DPO enh0m+enh3m models and AV1. The sequences are from YouTube UGC dataset.[31] Videos marked with an asterisk (*) are excluded from the averages and are discussed separately.

aomenc cpu setting	cpu=5			cpu=3		
Sequence	SSIM (%)	AH-VMAF (%)	VMAF (%)	SSIM (%)	AH-VMAF (%)	VMAF (%)
LiveMusic_1080P-14af	3.39	-9.43	-21.35	2.19	-6.86	-20.56
LiveMusic_1080P-157b*	25.69	-10.17	-35.85	-7.09	-7.51	-34.03
LiveMusic_1080P-1ace	-0.68	-0.19	-18.57	-0.66	-0.48	-19.68
LiveMusic_1080P-21dd	-1.13	-9.93	-27.05	-0.69	-7.02	-26.09
LiveMusic_1080P-28fe	-1.97	-8.27	-24.90	-0.40	-10.59	-26.33
LiveMusic_1080P-2930	-2.18	-9.51	-20.25	-1.84	-7.26	-17.93
LiveMusic_1080P-2b7a	-2.56	-10.61	-27.47	-2.19	-8.59	-25.79
LiveMusic_1080P-2f7f	-2.38	-8.16	-21.41	-1.83	-7.17	-23.00
LiveMusic_1080P-3549	-11.52	-7.47	-25.44	-8.19	-3.86	-24.51
LiveMusic_1080P-3e1a*	-0.12	1.47	-46.58	75.00	-0.01	-49.45
LiveMusic_1080P-3f95	-1.23	-4.48	-36.95	-1.13	-1.48	-35.83
LiveMusic_1080P-48d5	-7.40	-8.17	-19.00	-5.12	-4.76	-17.37
LiveMusic_1080P-514e	-5.10	-17.21	-18.77	-5.46	-29.24	-60.38
LiveMusic_1080P-51f6	3.14	-8.37	-20.02	4.48	-8.04	-20.11
LiveMusic_1080P-541f	-4.98	-11.88	-32.22	1.31	-8.09	-29.88
LiveMusic_1080P-59b3*	18.72	33.19	-18.76	-13.66	27.10	-68.16
LiveMusic_1080P-6b1c	-1.24	-3.24	-23.88	-1.43	-2.46	-24.00
LiveMusic_1080P-6bbe	-1.32	-9.65	-17.18	-0.34	-4.71	-13.52
LiveMusic_1080P-6d1a	-0.13	-7.28	-31.64	0.74	-4.91	-29.44
LiveMusic_1080P-6fe2	0.08	-7.79	-26.34	0.52	-5.45	-24.10
LiveMusic_1080P-77e8	3.83	-15.32	-75.00	2.11	-14.16	-75.00
LiveMusic_1080P-7948*	75.00	75.00	75.00	75.00	75.00	75.00
LiveMusic_1080P-7ead	4.51	-11.49	-14.53	11.09	-16.09	-29.39
Sports_1080P-0063	-4.98	-22.68	-43.05	-5.58	-14.62	-37.08
Sports_1080P-0640	-0.08	-9.97	-19.31	-0.72	-9.17	-18.75
Sports_1080P-08e1	-5.24	-3.58	-20.46	-3.13	-1.25	-18.03
Sports_1080P-0d0c	-4.66	-7.64	-21.60	-3.96	-7.08	-22.64
Sports_1080P-15d1	-6.48	-11.20	-25.90	-3.87	-9.16	-25.28
Sports_1080P-19d8	39.70	-7.81	-48.45	10.73	-5.35	-47.25
Sports_1080P-1ae3	-3.36	-13.16	-32.14	-3.03	-11.20	-30.77
Sports_1080P-1bf7	-5.43	-10.34	-31.13	-2.32	-8.76	-30.60
Sports_1080P-1d78	-5.89	-19.73	-38.08	-4.98	-14.76	-34.08
Sports_1080P-241e	-6.41	-4.67	-25.30	-6.46	-1.31	-19.65
Sports_1080P-2524	-0.66	-8.41	-31.41	-1.05	-5.02	-28.95
Sports_1080P-28a6	2.34	2.92	-37.85	0.10	7.10	-46.34
Sports_1080P-2a21	-2.57	-11.85	-23.31	0.95	-7.93	-21.97
Sports_1080P-3a3b	-3.59	-12.97	-26.17	-2.55	-9.82	-24.43
Sports_1080P-3db7	-0.76	-12.49	-34.76	0.43	-7.89	-23.75
Sports_1080P-3eb0	-1.27	-10.82	-45.46	-0.88	-8.92	-39.19
Sports_1080P-43e2*	75.00	0.15	-40.89	75.00	2.22	-38.29
Sports_1080P-46ed	-1.67	-13.57	-35.56	0.31	-8.41	-31.95
Sports_1080P-47e9	1.45	0.21	-44.07	-24.96	4.70	-40.03
Sports_1080P-4978	0.57	-4.59	-27.85	3.01	-0.47	-27.00
Sports_1080P-49c5	0.26	-1.84	-25.54	0.17	-4.41	-28.44
Sports_1080P-4e05	-3.90	-16.87	-37.10	1.33	-11.97	-33.73
Sports_1080P-53a0	-1.90	-14.71	-36.07	-2.68	-11.07	-35.4
Sports_1080P-5d25	-4.60	-23.34	-39.10	-5.4	-19.44	-37.46
Sports_1080P-6571	-1.34	-10.51	-38.58	-0.82	-6.64	-37.54
Sports_1080P-6710	-2.16	-8.13	-46.65	-1.80	-5.96	-48.86
Sports_1080P-679d	0.22	-1.03	-32.40	0.64	-1.62	-49.90
Sports_1080P-7203*	3.74	75.00	75.00	2.49	75.00	75.00
Sports_1080P-7584	-1.53	-9.00	-19.95	-1.08	-6.71	-18.44
Sports_1080P-76a2	-2.77	-10.92	-27.21	-2.93	-9.53	-26.70
Sports_1080P-78fa	-0.25	-9.68	-29.29	3.07	-8.61	-26.82
Sports_1080P-7b51	-3.82	-16.05	-29.41	-3.12	-13.97	-28.44
Sports_1080P-7dba	-5.64	-1.84	-32.01	-3.82	1.37	-28.98
Average	**-1.31**	**-9.49**	**-30.14**	**-1.42**	**-7.38**	**-30.23**
Av. AH-VMAF & VMAF	–	**-19.82**		–	**-18.81**	
Average of all three	**-13.65**			**-13.01**		

Table 5: Average BD-rate results (%) over VMAF, AH-VMAF and SSIM under the use of sharpening filters prior to encoding. For x264, an FFmpeg sharpening filter recipe is used. For aomenc, the `-tune=vmaf_with_preprocessing` flag was used.

SSIM				
Dataset	sharp+x264 slow	sharp+x264 veryslow	sharp+aomenc cpu=5	sharp+aomenc cpu=3
XIPH+CDVL	21.56	22.61	18.83	22.70
UGC	45.46	45.64	27.45	28.62
AH-VMAF				
XIPH+CDVL	4.86	8.22	9.41	-19.48
UGC	16.49	18.71	21.48	26.06
VMAF				
XIPH+CDVL	-34.21	-34.04	-30.04	-27.08
UGC	-32.98	-32.97	-32.36	-31.83
Average over SSIM, AH-VMAF and VMAF				
XIPH+CDVL	-2.60	-1.07	-0.60	-7.95
UGC	9.66	10.46	5.52	7.62

Table 6: Average BD-rate results (%) over VMAF, AH-VMAF and SSIM when transitioning between different encoders. Interpretation example: Row 'x264 veryslow' and column 'x264 slow' shows the BD-rate increase of 7.75% incurred when switching from x264 veryslow preset to the x264 slow preset.

From Baseline	To Target			
	x264 slow	x264 veryslow	aomenc cpu=5	aomenc cpu=3
x264 slow	0.00	-7.07	-33.90	-40.96
x264 veryslow	**7.75**	0.00	-29.17	-36.72
aomenc cpu=5	51.16	**43.63**	0.00	-10.88
aomenc cpu=3	60.02	55.43	**12.45**	0.00

Table 7: Average BD-rate results (%) over VMAF, AH-VMAF and SSIM when transitioning between different encoders and encoders with our proposed DPO enh0m+enh3m precoding models. Interpretation example: Row 'x264 veryslow' and column 'DPO+x264 slow' shows the BD-rate reduction of 7.89% incurred when switching from x264 'veryslow' preset to the DPO+x264 'slow' preset.

From Baseline	To Target			
	DPO+x264 slow	DPO+x264 veryslow	DPO+aomenc cpu=5	DPO+aomenc cpu=3
x264 slow	-14.35	-20.00	-41.42	-47.5
x264 veryslow	**-7.89**	-14.03	-37.28	-43.81
aomenc cpu=5	32.50	**24.46**	-13.33	-21.40
aomenc cpu=3	45.70	38.00	**-2.94**	-11.62

Table 8: Average speed ratio: $\frac{\text{Target\_Speed}}{\text{Baseline\_Speed}}$ (numbers below 1.00 indicate slowdown) when transitioning between different encoders. Interpretation example: Row 'x264 veryslow' and column 'x264 slow' shows that switching from x264 'veryslow' to 'slow' allows for 2-fold increase in encoding speed.

From Baseline	To Target			
	x264 slow	x264 veryslow	aomenc cpu=5	aomenc cpu=3
x264 slow	1.00	0.50	0.03	0.02
x264 veryslow	**2.00**	1.00	0.07	0.04
aomenc cpu=5	29.02	**14.50**	1.00	0.53
aomenc cpu=3	54.55	27.25	**1.88**	1.00

slowdown of encoding speed to 50%, but at the same time increases the bitrate reduction from 7.07% to 20%, which can make such a switch very worthwhile. Overall Table 6-Table 8 show how our approach helps bridge the gap between encoder complexity presets, or even between encoding standards generations. While the DPO realization itself requires GPU or multi-CPU with support for fast quantized neural network inference in order to run in real time, *our models only need to apply to the content once*, and multiple encoding runs can be carried out. For example, its complexity can become negligible in comparison to the complexity of the 50 or more encodings per video needed in order to produce the dynamic optimizer results reported in this paper. This single-pass property of DPO reduces the implementation overhead versus conventional encoding. We intend to reduce the runtime complexity with further speed optimizations and a content-adaptive selection between models that will have minimal impact in the obtained results, or can even provide for further bitrate gains. We have noticed that the deployment pace of such optimizations is significantly faster than optimizing hand-crafted code of encoders, since they are carried out via training, finetuning, prunning and quantization/approximation experiments that are data-driven and operate in a semi-automatic manner.

6. CONCLUSION

We propose deep perceptual optimization as the means of generating a perceptually enhanced and rate-controlled representation of each input frame via learnable preprocessing or 'precoding'. Our proposed DPO framework models the building blocks of a standard video encoder in order to optimize the precoding for rate, distortion and perceptual quality in an end-to-end differentiable manner for backpropagation. At inference, only the precoder is deployed and prepended to carry out a single pass through each frame prior to any standard encoder targeting any bitrate. Our results show that our learned precoding offers controllable gains on both standard distortion and perceptual-oriented metrics, such as VMAF and SSIM, when evaluated on two codec generations used in streaming systems. This allows for complexity-bitrate-quality trade-offs that go beyond those offered by tunable encoder recipes in standards, thereby making transition between older and newer standards more seamless, or, conversely, making the transition to a newer and more complex encoder significantly more beneficial in terms of quality reported by standard metrics. The full set of libvmaf JSON measurements and Matlab/Octave code to calculate the optimal rate-quality convex hull per sequence and obtain the BD-rates of this paper is available online at https://www.isize.co/spie2020.zip as a 3.8GB file deflating to a 29GB cache of 296K measurement files. The main script to compute BD-rates between the results of any two folders is `json_sequence_folders_comp_v1.m`.

REFERENCES

[1] Huynh-Thu, Q. and Ghanbari, M., "Scope of validity of PSNR in image/video quality assessment," *Electronics letters* **44**(13), 800–801 (2008).

[2] Li, Z., Aaron, A., Katsavounidis, I., Moorthy, A., and Manohara, M., "Toward a practical perceptual video quality metric," *The Netflix Tech Blog* **6** (2016).

[3] Talebi, H. and Milanfar, P., "Nima: Neural image assessment," *IEEE Transactions on Image Processing* **27**(8), 3998–4011 (2018).

[4] Zhang, W., Ma, K., Yan, J., Deng, D., and Wang, Z., "Blind image quality assessment using a deep bilinear convolutional neural network," *IEEE Transactions on Circuits and Systems for Video Technology* (2018).

[5] Kim, J. and Lee, S., "Deep learning of human visual sensitivity in image quality assessment framework," in [*Proceedings of the IEEE conference on computer vision and pattern recognition*], 1676–1684 (2017).

[6] Katsavounidis, I. and Guo, L., "Video codec comparison using the dynamic optimizer framework," in [*Applications of Digital Image Processing XLI*], **10752**, 107520Q, International Society for Optics and Photonics (2018).

[7] Ballé, J., Laparra, V., and Simoncelli, E. P., "End-to-end optimized image compression," *arXiv preprint arXiv:1611.01704* (2016).

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

[9] Rippel, O. and Bourdev, L., "Real-time adaptive image compression," in [*Proceedings of the 34th International Conference on Machine Learning-Volume 70*], 2922–2930, JMLR. org (2017).

[10] Theis, L., Shi, W., Cunningham, A., and Huszár, F., "Lossy image compression with compressive autoencoders," *arXiv preprint arXiv:1703.00395* (2017).
[11] Lu, G., Ouyang, W., Xu, D., Zhang, X., Cai, C., and Gao, Z., "Dvc: An end-to-end deep video compression framework," in [*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*], 11006–11015 (2019).
[12] Djelouah, A., Campos, J., Schaub-Meyer, S., and Schroers, C., "Neural inter-frame compression for video coding," in [*Proceedings of the IEEE International Conference on Computer Vision*], 6421–6429 (2019).
[13] Choi, Y., El-Khamy, M., and Lee, J., "Variable rate deep image compression with a conditional autoencoder," in [*Proceedings of the IEEE International Conference on Computer Vision*], 3146–3154 (2019).
[14] Rippel, O., Nair, S., Lew, C., Branson, S., Anderson, A. G., and Bourdev, L., "Learned video compression," in [*Proceedings of the IEEE International Conference on Computer Vision*], 3454–3463 (2019).
[15] De Cock, J., Mavlankar, A., Moorthy, A., and Aaron, A., "A large-scale video codec comparison of x264, x265 and libvpx for practical vod applications," in [*Applications of Digital Image Processing XXXIX*], **9971**, 997116, International Society for Optics and Photonics (2016).
[16] Agustsson, E., Minnen, D., Johnston, N., Balle, J., Hwang, S. J., and Toderici, G., "Scale-space flow for end-to-end optimized video compression," in [*Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*], 8503–8512 (2020).
[17] Grange, A., Norkin, A., Chen, C., Chiang, C.-H., Mukherjee, D., Su, H., Bankoski, J., Valin, J.-M., Han, J., Trudeau, L., et al., "An overview of core coding tools in the av1 video codec," (2018).
[18] Chiang, C.-H., Han, J., and Xu, Y., "A multi-pass coding mode search framework for AV1 encoder optimization," in [*2019 Data Compression Conference (DCC)*], 458–467, IEEE (2019).
[19] Su, H., Chen, M., Bokov, A., Mukherjee, D., Wang, Y., and Chen, Y., "Machine learning accelerated transform search for AV1," in [*2019 Picture Coding Symposium (PCS)*], 1–5, IEEE (2019).
[20] Bourtsoulatze, E., Chadha, A., Fadeev, I., Giotsas, V., and Andreopoulos, Y., "Deep video precoding," *IEEE Transactions on Circuits and Systems for Video Technology* (2019).
[21] Afonso, M., Zhang, F., and Bull, D. R., "Video compression based on spatio-temporal resolution adaptation," *IEEE Transactions on Circuits and Systems for Video Technology* **29**(1), 275–280 (2018).
[22] Blau, Y. and Michaeli, T., "The perception-distortion tradeoff," in [*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*], 6228–6237 (2018).
[23] Zvezdakova, A., Zvezdakov, S., Kulikov, D., and Vatolin, D., "Hacking VMAF with video color and contrast distortion," in [*Proc. of the 29th International Conference on Computer Graphics and Vision (GraphiCon 2019). CEUR Workshop Proceedings*], **2485**, 53–57 (2019).
[24] Li, Z., Bampis, C., Novak, J., Aaron, A., Swanson, K., Moorthy, A., and Cock, J., "VMAF: The journey continues," *Netflix Technology Blog* (2018).
[25] Yu, F. and Koltun, V., "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122* (2015).
[26] Malvar, H. S., Hallapuro, A., Karczewicz, M., and Kerofsky, L., "Low-complexity transform and quantization in H.264/AVC," *IEEE Transactions on circuits and systems for video technology* **13**(7), 598–603 (2003).
[27] Marpe, D., Schwarz, H., and Wiegand, T., "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).
[28] Talebi, H. and Milanfar, P., "Learned perceptual image enhancement," in [*2018 IEEE International Conference on Computational Photography (ICCP)*], 1–13, IEEE (2018).
[29] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L., "Imagenet: A large-scale hierarchical image database," in [*2009 IEEE conference on computer vision and pattern recognition*], 248–255, Ieee (2009).
[30] Wang, Z., Simoncelli, E. P., and Bovik, A. C., "Multiscale structural similarity for image quality assessment," in [*The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*], **2**, 1398–1402, Ieee (2003).
[31] Wang, Y., Inguva, S., and Adsumilli, B., "YouTube UGC dataset for video compression research," in [*2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*], 1–5, IEEE (2019).
[32] Bjontegaard, G., "Improvements of the BD-PSNR model," in [*ITU-T SG16/Q6, 35th VCEG Meeting, Berlin, Germany, July, 2008*], (2008).