Deep Video Precoding

Eirina Bourtsoulatze, Aaron Chadha, Ilya Fadeev, Vasileios Giotsas, and Yiannis Andreopoulos

Abstract—Several groups worldwide are currently investigating how deep learning may advance the state-of-the-art in image and video coding. An open question is how to make deep neural networks work in conjunction with existing (and upcoming) video codecs, such as MPEG H.264/AVC, H.265/HEVC, VVC, Google VP9 and AOMedia AV1, AV2, as well as existing container and transport formats, without imposing any changes at the client side. Such compatibility is a crucial aspect when it comes to practical deployment, especially when considering the fact that the video content industry and hardware manufacturers are expected to remain committed to supporting these standards for the foreseeable future.

We propose to use deep neural networks as precoders for current and future video codecs and adaptive video streaming systems. In our current design, the core precoding component comprises a cascaded structure of downscaling neural networks that operates during video encoding, prior to transmission. This is coupled with a precoding mode selection algorithm for each independently-decodable stream segment, which adjusts the downscaling factor according to scene characteristics, the utilized encoder, and the desired bitrate and encoding configuration. Our framework is compatible with all current and future codec and transport standards, as our deep precoding network structure is trained in conjunction with linear upscaling filters (e.g., the bilinear filter), which are supported by all web video players. Extensive evaluation on FHD (1080p) and UHD (2160p) content and with widely-used H.264/AVC, H.265/HEVC and VP9 encoders, as well as a preliminary evaluation with the current test model of VVC (v.6.2rc1), shows that coupling such standards with the proposed deep video precoding allows for 8% to 52% rate reduction under encoding configurations and bitrates suitable for video-on-demand adaptive streaming systems. The use of precoding can also lead to encoding complexity reduction, which is essential for cost-effective cloud deployment of complex encoders like H.265/HEVC, VP9 and VVC, especially when considering the prominence of high-resolution adaptive video streaming.

Index Terms—video coding, neural networks, downscaling, upscaling, adaptive streaming, DASH/HLS

I. Introduction

IN JUST a few years, technology has completely overhauled the way we consume television, feature films and other prime content. For example, Ofcom reported in July 2018 that there are now more UK subscriptions to Netflix, Amazon and NOW TV than to traditional pay TV services.[1] The proliferation of over-the-top (OTT) streaming content has been matched by an appetite for high-resolution content. For example, 50% of the US homes will have UHD/4K TVs by 2020. At the same time, costs of 4K camera equipment have been falling rapidly. Looking ahead, 8K TVs were introduced

[1]https://www.ofcom.org.uk/about-ofcom/latest/media/media-releases/2018/streaming-overtakes-pay-tv

at the 2018 CES by several major manufacturers and several broadcasters announced they will begin 8K broadcasts in time for the 2020 Olympic games in Japan. Alas, for most countries, even the delivery of FHD (1080p) content is still plagued by broadband infrastructure problems.

To get round this problem, OTT content providers resort to adaptive streaming technologies, such as Dynamic Adaptive Streaming over HTTP (DASH) and HTTP Live Streaming (HLS), where the streaming server is offering bitrate/resolution ladders via the so-called "manifest" file [1], [2]. This allows the client device to switch to a range of lower resolutions and bitrates when the connection bandwidth does not suffice to support the high-quality/full-resolution video bitstream [1]. In order to produce the bitrate/resolution ladders, high-resolution frames are downscaled to lower resolutions, with the frame dimensions being reduced with decreased bitrate. Within all current adaptive streaming systems, this is done using standard downscaling filters, such as the bicubic filter, and the chosen resolution per bitrate stays constant throughout the content's duration and is indicated in the manifest file. At the client side, if the chosen bitrate corresponds to low-resolution frames, these frames are upscaled after decoding to match the resolution capabilities of the client's device. Unfortunately, the impact on visual quality from the widely-used bicubic downscaling and the lack of dynamic resolution adaptation per bitrate can be quite severe [3]. In principle, this could be remedied by post-decoding learnable video upscaling solutions similar to learnable super-resolution techniques for still images [4], [5]. However, their deployment requires substantial changes to the client device, which are usually too cumbersome and complex to make in practice (a.k.a., the hidden technical debt of machine learning [6]). For example, convolutional neural network (CNN) based upscalers with tens of millions of parameters cannot be supported by mainstream CPU-based web browsers that support DASH and HLS video playback.

A. Precoding for Video Communications

Precoding has been initially proposed for MIMO wireless communications as the means to preprocess the transmitted signal and perform transmit diversity [7]. Precoding is similar to channel equalization, but the key difference is that it shapes (precodes) the signal according to the operation of the utilized receiver *prior* to channel coding. While channel equalization aims to minimize channel errors, a precoder aims to minimize the error in the receiver output.

In this paper, we introduce the concept of precoding for adaptive video streaming. As illustrated in Fig. 1, precoding for video is done by preprocessing the input video frames of each independently-decodable group of pictures (GOP) prior
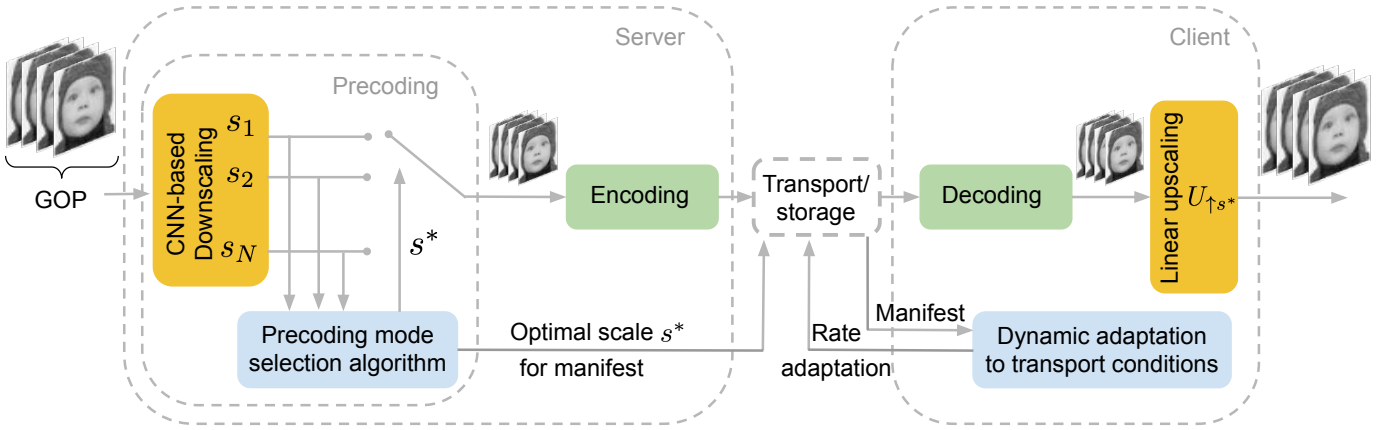
Fig. 1. Proposed deep video precoding framework. The precoding module performs dynamic resolution adaptation during encoding, prior to streaming. The optimal scale factor is chosen by our adaptive precoding mode selection algorithm which operates per GOP, bitrate and codec and is passed to the server-based manifest file that points all adaptive streaming clients to the available content chunks (GOPs) and their bitrate and resolution settings.

to standard encoding, while allowing a standard video decoder and player to decode and display them without requiring any modifications. The key idea of precoding is to minimize the distortion at the output of the client's player. To achieve this, we leverage on the support for multiple resolutions at the player side and introduce a multi-scale precoding convolutional neural network (CNN) that progressively downscales input high-resolution frames over multiple scale factors. A mode selection algorithm then selects the best precoding mode (i.e., resolution) to use per GOP based on the GOP frame content, bitrate and codec characteristics. The precoding CNN is designed to compact information in such a way that the aliasing and blurring artifacts generated during linear upscaling are mitigated. This is because, in the vast majority of devices, video upscaling is performed by means of linear filters. Thus, our proposed deep video precoding solution focuses on matching standard video players' built-in linear upscaling filter implementations, such as the bilinear filter[2].

Our experiments with standard FHD (1080p) and UHD (2160p) test content from the XIPH repository and well-established implementations of H.264/AVC, H.265/HEVC and VP9 encoders show that, by using deep precoding modes for downscaling, we can significantly reduce the distortion of video playback compared to conventional downscaling filters and fixed downscaling modes used in standard DASH/HLS streaming systems. We further demonstrate through extensive experimentation that the proposed adaptive precoding mode selection achieves 8% to 52% bitrate reduction for FHD and UHD content encoding at typical bitrate ranges used in commercial deployments. An important effect of the proposed precoding is that video encoding can be accelerated, since

many GOPs tend to be shrunk by the precoder to only 6%-64% of their original size, depending on the selected downscaling factor. This is beneficial when considering cloud deployments of such encoders, especially, in view of upcoming standards with increased encoding complexity.

In summary, our contributions are as follows:

- the concept of deep precoding for video delivery is introduced as the means of enhancement of the rate-distortion characteristics of any video codec without requiring any changes on the client/decoder side;
- a multi-scale precoding CNN is proposed, which downscales high-resolution frames over multiple scale factors and is trained to mitigate the aliasing and blurring artifacts generated by standard linear upscaling filters;
- an adaptive precoding mode selection algorithm is proposed, which adaptively selects the optimal resolution prior to encoding.

It is important to emphasize that deep video precoding is a *source encoding* optimization framework carried out at the server side, prior to transmission, in order to optimally "shape" the input signal by deriving the best downscaled representation according to input GOP segment, bitrate, codec and upscaling capabilities at the receiver, without considering transport conditions. Fig. 1 illustrates how our proposed deep video precoding can be used in conjunction with client-driven adaptive streaming systems like the widely-deployed DASH/HLS standards. As shown in Fig. 1, precoding for each content, codec and target bitrate is independent of how DASH/HLS players will switch between encoding ladders to adapt the video bitrate to the transport bandwidth, latency and buffer conditions. That is, once the DASH/HLS-compliant stream is produced by our approach for each encoding bitrate and the manifest file is created with the corresponding bitrate and resolution information per video segment, adaptive bitrate streaming, stream caching and stream switching mechanisms

---

[2]Despite an abundance of possible upscaling filters, in order to remain efficient over a multitude of client devices, most web browsers only support the bilinear filter for image and video upscaling in YUV colorspace, e.g., see the Chromium source code that uses libyuv (https://chromium.googlesource.com).

that cope with bitrate and channel fluctuations can operate as usual. The key difference in our case is that the client receives, decodes and upscales bespoke representations created by the proposed precoder. Hence, our proposal for deep video precoding is done in its entirety during content encoding and remains agnostic to the transport conditions experienced during each individual video bitstream delivery.

B. Paper Organization

The remainder of this paper is organized as follows. In Sec. II, we review related work. The proposed multi-scale deep precoding network architecture design, loss function, and implementation and training details are presented in Sec. III. The precoding mode selection algorithm is presented in Sec. IV. Experimental results are given in Sec. V, and Sec. VI concludes the paper.

II. RELATED WORK

Content-adaptive encoding has emerged as a popular solution for quality or bitrate gains in standards-based video encoding. Most commercial providers have already demonstrated content-adaptive encoding solutions, typically in the form of bitrate adaptation based on combinations of perceptual metrics, i.e., lowering the encoding bitrate for scenes that are deemed to be simple enough for a standard encoder to process. Such solutions can also be extended to other encoder parameter adaptations, and their essence is in the coupling of a visual quality profile to a pre-cooked encoder-specific tuning recipe.

A. Resolution Adaptation in Image and Video Coding

It has been known for some time that reducing the input resolution in image or video coding can improve visual quality for lower bitrates as the encoder operates better at the "knee" of its rate-distortion curve [8]. Starting from non-learnable designs, Tsaig *et al.* [9] explored the design of optimal decimation and interpolation filters for block image coders like JPEG and showed that low-bitrate image coding between 0.05 bits-per-pixel (bpp) to 0.35 bpp benefits from such designs. Kopf *et al.* [10] proposed a content-adaptive method, wherein filter kernel coefficients are adapted with respect to image content. Oztireli *et al.* [11] proposed an optimization framework to minimize structural similarity between the nearest-neighbor upsampled low-resolution image and the high-resolution image. Recently, Katsavounidis *et al.* [3], [12] proposed the notion of the dynamic optimizer in video encoding: each scene is downscaled to a range of resolutions and is subsequently compressed to a range of bitrates. After upscaling to full resolution, the convex hull of bitrates/qualities is produced in order to select the best operating resolution for each video segment. Quality can be measured with a wide range of metrics, ranging from simple peak signal to noise ratio (PSNR) to complex fusion-based metrics like the video multimethod assessment fusion (VMAF) metric [13]. While Bjontegaard distortion-rate (BD-rate) [14] gains of 30% have been shown in experiments for H.264/AVC and VP9, the dynamic optimizer requires very

significant computational resources, while it still uses non-learnable downscaling filters. A learnable CCN-based downscaling method for image compaction was proposed by Li *et al.* [15], where the downscaling CNN is trained jointly with either a fixed linear upscaling filter or a trainable upscaling CNN. While that work also explored the concept of learnable downscaling, it is designed for fixed-ratio downscaling and does not provide content, bitrate and codec adaptivity as our proposed deep precoding approach. However, it forms an important learnable downscaling framework that we use as one of the benchmarks for our work.

B. Super-resolution Methods

Overall, while the above methods have shown the possibility of rate saving via image and video downscaling, they have not managed to significantly outperform classical bicubic downscaling within the context of practical encoding. This has led most researchers and practitioners to conclude that downscaling with bicubic or Lanczos filters is the best approach, and instead the focus has shifted on upscaling solutions at the client (i.e., decoder) side that learn to recover image detail assuming such downscaling operators. For example, Georgis et al. [16] proposed backprojection-based upscaling that is tailored to Gaussian kernel downscaling and showed that such approaches can be beneficial for FHD and UHD/4K encoding with H.264/AVC and HEVC up to 10mbps and 5mbps, respectively. The majority of recent works in this field consider CNN-based upscaling methods. This has been largely motivated by the success of deep CNN architectures for single image super-resolution, that have set the state-of-the-art, with recent architectures like VDSR [17], EDSR [18], FSRCNN [19], DRCN [20] and DBPN [4] achieving several dB higher PSNR in the luminance channel of standard image benchmarks for lossless image upscaling. Thus, Afonso *et al.* propose a spatio-temporal resolution adaptation where a CNN-based super-resolution model is used to reconstruct full-resolution content [21]. Li *et al.* [22] introduce the block adaptive resolution coding framework for intra frame coding, where each block within a frame is either downscaled or coded at original resolution and then upscaled with a trained CNN at the decoder side. This concept was later extended to include P and B frames as well [23]. Differently from the previous methods that operate in the pixel domain, Liu *et al.* perform down and upsampling in the residue domain and design upsampling CNN for residue super resolution (SR) with the help of the motion compensated prediction signal [24]. However, regardless of the domain where they operate, the common principle of all these works is that they use hand-crafted filters for downscaling and perform upscaling at the decoder side using codec-tailored CNN models integrated in the coding standard which requires modifications to the codec.

C. Neural-network Representation and Coding Methods

While most research efforts have focused on learning optimal upscaling filters [25], inspired by the success of autoencoders for image compression [26], [27], some recent works revise the problem of joint downscaling and upscaling

using deep CNN-based methods. Shocher *et al.* [28] recently proposed an upscaling method using deep learning, which trains an image-specific CNN with high and low-resolution pairs of patches extracted from the test image itself. Weber *et al.* [29] used convolutional filters to preserve important visual details, and Hou *et al.* [30] recently proposed a deep perceptual loss based method. Kim *et al.* [31] proposed an image downscaling/upscaling method using a deep convolutional autoencoder, and achieved state-of-the-art results on standard image benchmarks. An end-to-end image compression framework for low bitrates compatible with existing image coding standards was introduced in [32]. It comprises two CNNs: one for image compaction prior to encoding and one for post-processing after decoding. Adaptive spatio-temporal decomposition prior to encoding, followed by CNN-based spatio-temporal upscaling after decoding was proposed by Afonso *et al.* and was validated with H.265/HEVC encoding [21]. Finally, Wave One recently proposed video encoding with deep neural networks [33] and demonstrated quality gains against a conventional video encoder without B frames, and focusing on very-high bitrate encoding (20mbps or higher for FHD). While these are important achievements, most of these proposals are still outperformed by post-2013 video encoders, like HEVC and VP9, when utilized with their most advanced video buffering verifier (VBV) encoding configurations and appropriate constant rate factor tuning [34]. In addition, all these proposals require advanced GPU capabilities on both the client and the server side that cannot be supported by existing video players as they break away from current standards. Therefore, despite the significant advances that may be offered by all these methods in their future incarnations, they do not consider the stringent complexity and standards compatibility constraints imposed when dealing with adaptive video streaming under DASH or HLS-compliant clients like web browsers. Our work fills this gap by offering deep video precoding as the means to optimize existing video encoders with the entirety of the precoding process taking place on the server side and not requiring any change in the video transport, decoding and display side.

III. MULTI-SCALE PRECODING NETWORKS

While any downscaling method can be used with the video precoding framework of Fig. 1, to enhance the performance of our proposal in a data-driven manner, we introduce a multi-scale precoding neural network. The precoding network comprises a series of CNN precoding blocks that progressively downscale high resolution (HR) video frames over multiple scale factors. We design the precoding CNN to compact information such that a standard linear upscaler at the video player side will be able to recover in the best possible way. This is the complete opposite of recent image upscaling architectures that assume simple bicubic downscaling and an extremely complex super-resolution CNN architecture at the video player side. For example, EDSR [5] comprises over 40 million parameters and would be highly impractical on the client side for 30-60 frame-per-second (fps) FHD/UHD video.

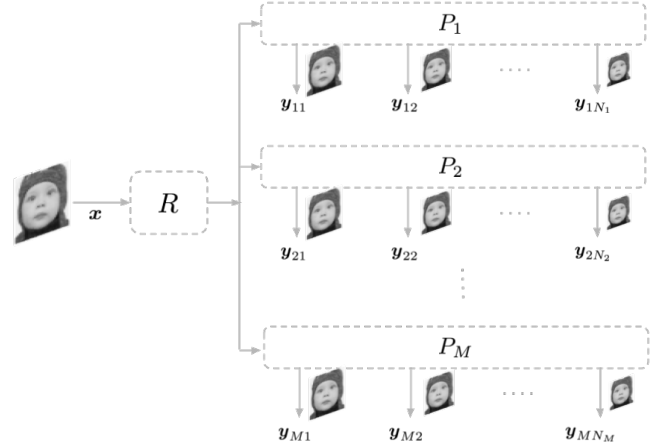In the following subsections, we describe the design of the proposed multi-scale precoding networks, including the network architecture, loss function, and details of implementation and training.

Fig. 2. The architecture of our multi-scale precoding network for video downscaling, comprising a root mapping R and precoding streams $P_1, P_2, \ldots P_M$. The luminance frame of each input video frame is downsampled to multiple lower resolutions by the precoding network at the server via the precoding streams.

A. Network Architecture

The overall architecture of the precoding network is depicted in Fig. 2. It consists of a "root" mapping R followed by M parallelized precoding streams P_m. The network progressively downscales individual luminance frames $\boldsymbol{x} \in \mathbb{R}^{H \times W}$ (where H and W are the height and width, respectively) over the scale factors in $\boldsymbol{S}$. Considering that the human eye is most sensitive to luma information, we intentionally process only the luminance (Y) channel with the precoding network and not the chrominance (Cb, Cr) channels, in order to avoid unnecessary computation. Dong *et al.* [35] support this claim empirically and, additionally, find that training a network on all three channels can actually worsen performance due to the network falling into a bad local minimum. We also note that this permits for chroma subsampling (e.g., YUV420) as the chrominance channels (Cb,Cr) are downscaled independently using the standard bicubic filter.

1) Root mapping: The root mapping R, illustrated in Fig 3, comprises two convolutional layers and extracts a set of high-dimensional feature maps $\boldsymbol{r} \in \mathbb{R}^{H \times W \times K}$ from the input $\boldsymbol{x}$, where K is the number of output channels of the root mapping. The root mapping R constitutes less abstract features (such as edges) that are common amongst all precoding streams and scale factors. Therefore, this module is shared between all precoding streams, which helps in reducing complexity.

2) Precoding stream: The extracted feature maps $\boldsymbol{r}$ are passed to the precoding streams. As depicted in Fig. 3, a precoding stream P_m comprises a sequence of N_m precoding blocks, which progressively downscale the input over a subset of N_m scale factors, $\boldsymbol{S}_m = \{s_{m1}, s_{m2}, \ldots, s_{mN_m}\} \subseteq \boldsymbol{S}$, where $1 < s_{m1} < s_{m2} < \cdots < s_{mN_m}$. The allocation of scale factors to a precoding stream is done in such a way that: *(i)* complexity is shared equally between streams for efficient
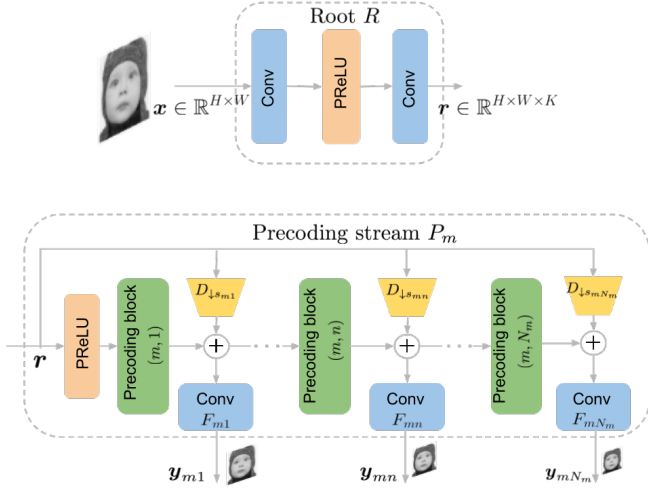
Fig. 3. Root mapping R and m-th precoding stream P_m. The root mapping extracts high-dimensional feature maps r and is shared by all precoding streams. The precoding stream P_m contains a sequence of precoding blocks and progressively downsamples the input high-resolution frames over a set of N_m scale factors.

parallel processing and *(ii)* the ratio of most of the consecutive pairs of scales $s_{m(n-1)}, s_{mn}$ within a stream is constant, i.e., $\alpha_{mn} = s_{mn}/s_{m(n-1)} = \alpha_m \in \mathbb{Z}^+$. Such construction of precoding streams exploits the inter-scale correlations among similar scales and enables parameter sharing within each precoding stream, thus further reducing the computational complexity. Most importantly, it renders our network amenable to standard encoding resolution ladders, such as those used in DASH/HLS streaming [2].

Given a precoding stream P_m, the n-th constituent precoding block receives a function of the output map $p_{m(n-1)} \in \mathbb{R}^{H/s_{m(n-1)} \times W/s_{m(n-1)} \times K}$ from the preceding block and outputs embedding $p_{mn} \in \mathbb{R}^{H/s_{mn} \times W/s_{mn} \times K}$. Notably, we utilize a global residual learning strategy, where we use a skip connection and perform a pixel-wise summation between the root feature maps r (pre-activation function and after linear downscaling to the correct resolution with a linear downscaling operator $D_{\downarrow s_{mn}}$) and p_{mn}. Similar global residual learning implementations have also been adopted by SR models [36]–[38]. In our case, our precoding stream effectively follows a pre-activation configuration [39] without batch normalization. We find empirically that convergence during training is generally faster with global residual learning, as the precoding blocks only have to learn the residual map to remove distortion introduced by downscaling operations.

The resulting downscaled feature map is finally mapped by a single convolutional layer F_{mn} to y_{mn}. Importantly, the sequence of precoding blocks only operates in a higher dimensional feature space without any heavy bottlenecks. It is then the job of convolutional layers F_{mn} to aggregate block outputs into single channel representations of the input x, downscaled by a factor of s_{mn}. For N_m precoding blocks and set of N_m scale factors S_m, the embedding stream outputs a set of N_m corresponding downscaled representations
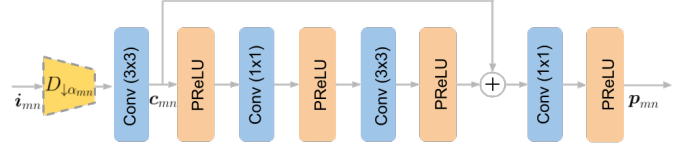

Fig. 4. Precoding block design, comprising a series of 3×3 and 1×1 convolutions. The linear downscaling operation $D_{\downarrow \alpha_{mn}}$ is only performed when the downscaling to the target resolution cannot be achieved via stride in the first convolutional layer. The linear mapping learned from the first layer (pre-activation function) is passed to the output of the second 3×3 convolutional layer (post-activation function) with a skip connection and pixel-wise summation.

$Y_m = \{y_{m1}, y_{m2}, \ldots, y_{mN_m}\}$ of the input x.

The output activations in Y_m are clipped (rescaled) between the minimum and maximum pixel intensities and each representation can, thus, be passed to the codec as a downscaled low resolution (LR) frame. These frames can then be individually upscaled to the original resolution using a linear upscaling $U_{\uparrow s}$ on the client side, such as bilinear, lanczos or bicubic, where $\uparrow s$ indicates upscaling by scale factor s. We refer to the upscaled frame generated from the downscaled frame y_{mn} as $\hat{x}_{mn}$ and denote the set of N_m upscaled representations of x as $\hat{X}_m = \{\hat{x}_{m1}, \hat{x}_{m2}, \ldots, \hat{x}_{mN_m}\}$.

3) Precoding block: Our precoding block, which constitutes the primary component of our network, is illustrated in Fig. 4. The precoding block consists of alternating 3×3 and 1×1 convolutional layers, where each layer is followed by a parametric ReLU (PReLU) [40] activation function. The 1×1 convolution is used as an efficient means for channel reduction in order to reduce the overall number of multiply-accumulates (MACs) for computation.

The n-th precoding block in the m-th precoding stream is effectively responsible for downscaling the original high resolution frame by a factor of s_{mn}. In order to maintain low complexity, it is important to downscale to the target resolution as early as possible. Therefore, we group all downsampling operations with the first convolutional layer in each precoding block. Denoting the input to the precoding block as i_{mn}, the output of the first convolutional layer C_{mn} of the precoding block as c_{mn} (as labelled in Figure 4) and spatial stride as k:

$$c_{mn} = \begin{cases} C_{mn}(i_{mn}; k = \alpha_{mn}), & \text{if } \alpha_{mn} \in \mathbb{Z}^+ \\ C_{mn}(D_{\downarrow \alpha_{mn}}(i_{mn}); k = 1), & \text{otherwise} \end{cases} \quad (1)$$

In other words, downscaling is implemented in the first convolutional layer with a stride if α_{mn} is an integer; otherwise we use a preceding linear downscaling operation $D_{\downarrow \alpha_{mn}}$ (bilinear or bicubic).

The aim of the precoding block is to reduce aliasing artifacts in a data-driven manner. Considering that the upscaling operations are linear and, therefore, heavily constrained, the network is asymmetric and the precoding structure can not simply learn to pseudo invert the upscaling. If that were to be the case, it would simply result in a traditional linear anti-aliasing filter, i.e., a lowpass filter that removes the high frequency

components of the image in a globally-tuned manner, such that the image can be properly resolved. Removing the high frequencies from the image leads to blurring. Adaptive deblurring is an inverse problem, as there is typically limited information in the blurred image to uniquely determine a viable input. As such, within our proposed precoding structure, we can respectively model the anti-aliasing and deblurring processes as a function composition of linear and non-linear mappings. As illustrated in Fig. 4, this is implemented with a skip connection between linear and non-linear paths, as utilized in ResNet [41] and its variants, again following the pre-activation structure. In order to ensure that the output of the non-linear path has full range $(-\infty, \infty)$, we can initialize the final PReLu before the skip connection such that it approximates an identity function.

B. Loss Function

Given the luminance channel of a ground-truth frame $x \in \mathbb{R}^{H \times W}$ (with Y ranging between 16-235 as per ITU-R BT.601 conversion), our goal is to learn the parameters for the root R and all precoding streams $P_1, P_2, \ldots P_M$. We denote the root module parameters as $\boldsymbol{\theta}$ and the parameters of the m-th precoding stream as $\boldsymbol{\phi}_m$. For the m-th precoding stream with downscaling over N_m scale factors and I training samples per batch, the composite loss function $\mathcal{L}_m$ can be defined as:

$$\mathcal{L}_m(\hat{\boldsymbol{X}}_m, \boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{\phi}_m) = \frac{1}{I} \sum_{i=1}^{I} \sum_{n=1}^{N_m} \left(\left| \hat{\boldsymbol{x}}_{mn}^{(i)} - \boldsymbol{x}^{(i)} \right| + \lambda \left| \nabla \hat{\boldsymbol{x}}_{mn}^{(i)} - \nabla \boldsymbol{x}^{(i)} \right| \right) \quad (2)$$

The first term represents the L1 loss between each generated upscaled frame $\hat{\boldsymbol{x}}_{mn} = U_{\uparrow s_{mn}}(\boldsymbol{y}_{mn})$ and the ground-truth frame $\boldsymbol{x}$, summed over all N_m scales, where $\boldsymbol{y}_{mn} = F_{m,n} \circ P_{m,\downarrow s_{mn}} \circ R(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{\phi}_m)$ and $P_{m,\downarrow s_{mn}}$ is the part of the m-th precoding stream that includes all precoding blocks up to the n-th block and is responsible for downscaling the input feature maps $\boldsymbol{r}$ by the scale factor s_{mn}. The second term represents the L1 loss between the first order derivatives of the generated high resolution and ground-truth frames. As the first order derivatives correspond to edge extraction, the second term acts as an edge preservation regularization for improving perceptual quality. We set the weight coefficient λ to 0.5 for all experiments; empirically, this was found to produce the best visual quality in output frames. Contrary to recent work [15], [31], we do not add a loss function constraint between the downscaling and upscaling as our upscaling is only linear and, therefore, already heavily constrains the downscaled frames. Most importantly, we do not include the codec in the training process and train end-to-end (i.e., without encoding/transcoding), such that the model does not learn codec specific dependencies and is able to generalize to multiple codecs. Finally, as we train the parallelized multi-scale precoding network over all streams synchronously, our final loss function is the summation of (2) over all M precoding streams:

$$\mathcal{L}(\hat{\boldsymbol{X}}, \boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{\phi}) = \sum_{m=1}^{M} \mathcal{L}_m(\hat{\boldsymbol{X}}_m, \boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{\phi}_m) \quad (3)$$

where $\hat{\boldsymbol{X}} = \hat{\boldsymbol{X}}_1 \cup \hat{\boldsymbol{X}}_2 \cup \cdots \cup \hat{\boldsymbol{X}}_M$ and $\boldsymbol{\phi} = \{\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_M\}$.

C. Implementation and Training Details

In our proposed multi-scale precoding network, we initialize all kernels using the method of Xavier intialization [42]. We use PReLU as the activation function, as indicated in Fig. 3 and 4. We use zero padding to ensure that all layers are the same size and downscaling is only controlled by a downsampling operation such as a stride or a linear downscaling filter. The root mapping R comprises a single 3×3 and 1×1 convolutional layers. We set the number of channels in all 1×1 and 3×3 convolutional layers to 4 and 8, respectively (excluding $F_{m,n}$, which uses a kernel size of 3×3 but with only a single output channel).

Our final parallelized implementation comprises three precoding streams P_1, P_2 and P_3, with the set of scale factors $\boldsymbol{S} \setminus \{1\}$ partitioned into three subsets: $\boldsymbol{S}_1 = \{4/3, 2, 4\}$, $\boldsymbol{S}_2 = \{3/2, 3, 6\}$ and $\boldsymbol{S}_3 = \{5/4, 5/2\}$. Collectively, these include all representative scale factors used in DASH/HLS streaming systems [2], as well as additional scale factors that offer higher flexibility in our adaptive mode selection algorithm. In terms of complexity, for a $1920 \times 1080 \times 4$ - dimensional feature map (assuming no downscaling), a single precoding block requires approximately only 1.33G MACs for downscaling and 640 parameters. Our final implementation requires only 3.38G MACs and 5.5K parameters over all scales per FHD input frame (1920×1080), including root mapping and all precoding streams.

We train the root module and all precoding streams end-to-end with linear upscaling, without the codec, on images from the DIV2K [43] training set. We train all models with the Adam optimizer with a batch size of 32 for 200k iterations. The initial learning rate is set as 0.001 and decayed by a factor of 0.1 at 100k iterations. We use data augmentation during training, by randomly flipping the images and train with a 120×120 random crops extracted from the DIV2K images. All experiments were conducted in Tensorflow on NVIDIA K80 GPUs. We do not use Tensorflow's built-in linear image resizing functions and rewrite all linear upscaling/downscaling functions from scratch, such that they match standard FFmpeg and OpenCV implementations.

IV. ADAPTIVE VIDEO PRECODING

Given the trained multi-scale precoding network, the mode selection algorithm operates on a per GOP, bitrate and codec basis. The goal of mode selection is to determine the optimal precoding scale factor for each GOP. While one can use operational rate-distortion (RD) models for H.264/AVC or H.265/HEVC [44] for evaluation of the best precoding modes, such models cannot encapsulate the complex and sequence-dependent RD behavior of each codec preset. On the other hand, exhaustive search solutions like the Netflix dynamic optimizer [3] require numerous highly-complex encodings per

bitrate and scale factor. This makes them impractical for high-volume/low-cost encoding systems. Our approach provides a middle ground between these two opposites by computing operational rate-distortion characteristics of the video encoder for each precoding mode and video segment in an efficient manner.

The precoding mode selection algorithm is outlined in Algorithm 1 and comprises three steps. The first step is to obtain the rate-distortion characteristics of each precoding mode (scale). Let $S = \{s_1, s_2, \ldots, s_N\}$ be the complete set of scale factors that may include the native (i.e., full) resolution. Each GOP segment g is precoded into all possible scales in S with the multi-scale precoding network. Let h_i denote the precoded version of GOP g using scale factor $s_i \in S$. All precoded versions of GOP g are then encoded with the video encoder's preset and encoding parameters. As such, per scale factor s_i, we obtain *a single* GOP encoding, which, after decoding and upscaling to the native resolution, produces *a single* rate-distortion point (R_i, D_i) on the RD plane.[3] This provides significant reduction of the required encodings versus approaches like the Netflix dynamic optimizer, which needs a convex hull of *several* rate-distortion points per scale factor s_i, i.e., several encodings per scale factor [3]. To accelerate this step even further and avoid unnecessary computational overhead, we introduce a "footprinting" process: instead of encoding all frames of the GOP, we perform selective encoding of only a few frames, e.g., keeping only every n-th frame in the GOP. This significantly speeds up the initial encoding step, especially if multiple precoding modes are considered, as n times fewer frames need to be precoded and encoded per scale factor. We store the set of tuples $R = \{(R_i, D_i, h_i, s_i)\}|_{i=1}^{|S|}$.

Once all the RD points are obtained, we start the pruning process. First, we eliminate all precoding modes, whose RD points do not provide for a monotonically decreasing RD curve. That is, for every precoding mode i, if there exists a precoding mode j, such that $R_j \leq R_i$ and $D_j < D_i$, the s_i precoding mode is pruned out. If, after this elimination procedure, the number of remaining precoding modes is greater than two, we further prune out precoding modes by eliminating those modes whose RD points do not lie on the convex hull of the remaining RD points [45], [3]. We refer to the set of pruned tuples as $R_{\text{pruned}} \subseteq R$. After the pruning stage, we re-encode the remaining precoded GOP representations h_i using constant bitrate (CBR) encoding. The bitrate used for the CBR encoding is equal to the average of the bitrates of all RD points remaining after the elimination step. After decoding and upscaling to the native resolution, we obtain a new set of tuples, R'_{pruned}, corresponding to the CBR encoding. This final step of the algorithm essentially remaps the RD points that remain after the pruning process to a common bitrate value, which is enforced by the CBR encoding. The optimal precoding mode s^* is then selected as the mode providing the lowest distortion among this new set of remapped RD points.

For illustration purposes, we demonstrate in Fig. 5a-5d the operation of the mode selection algorithm on the first

[3]We use mean squared error (MSE) to measure distortion. Even though other distortion measures could be used, MSE is fast to compute and is automatically provided by all encoders.

Algorithm 1 : Algorithm for adaptive mode selection

Input: GOP segment g, complete set of scale factors S
Output: optimal precoding mode s^*, optimal precoded version h^* of GOP segment g

1: **procedure** MODESELECTION(g, S)
    ▷ Step 1: Extract RD points for all precoding modes
2:    $R \leftarrow \{\}$
3:    **for each** $s \in S$ **do**
4:        $h \leftarrow$ PRECODE(g, s)
5:        $(e, R) \leftarrow$ ENCODE(h, preset, params)
6:        $\hat{h} \leftarrow$ DECODE(e)
7:        $\hat{g} \leftarrow$ UPSCALE($\hat{h}$)
8:        $D \leftarrow \|\hat{g} - g\|^2$
9:        $R \leftarrow R \cup \{(R, D, h, s)\}$
10:    **end for**
    ▷ Step 2: Prune out RD points
11:    $R_{\text{sorted}} \leftarrow ((R_i, D_i, h_i, s_i) \in R | R_i > R_{i-1})_{i=1}^{|S|}$
12:    $R_{\text{pruned}} \leftarrow \{R_{\text{sorted}}[1]\}; D_{\text{ref}} \leftarrow D_1$
13:    **if** $|S| > 1$ **then**
14:        **for** $i = 2$ **to** $|S|$ **do**
15:            $(R_i, D_i) \leftarrow R_{\text{sorted}}[i]$
16:            **if** $D_i < D_{\text{ref}}$ **then**
17:                $R_{\text{pruned}} \leftarrow R_{\text{pruned}} \cup \{(R_i, D_i, h_i, s_i)\}$
18:                $D_{\text{ref}} \leftarrow D_i$
19:            **end if**
20:        **end for**
21:    **end if**
22:    **if** $|R_{\text{pruned}}| > 2$ **then**
23:        $R_{\text{pruned}} \leftarrow$ CONVEXHULL(R_{pruned})
24:    **end if**
    ▷ Step 3: Re-encode remaining RD points with CBR
25:    $R'_{\text{pruned}} \leftarrow \{\}$
26:    **for each** $(R, D, h, s) \in R_{\text{pruned}}$ **do**
27:        $(e, R) \leftarrow$ ENCODE(h, preset, params, 'CBR')
28:        $\hat{h} \leftarrow$ DECODE(e)
29:        $\hat{g} \leftarrow$ UPSCALE($\hat{h}$)
30:        $D \leftarrow \|\hat{g} - g\|^2$
31:        $R'_{\text{pruned}} \leftarrow R'_{\text{pruned}} \cup \{(R, D, h, s)\}$
32:    **end for**
33:    $(R^*, D^*, h^*, s^*) \leftarrow \min_D(R'_{\text{pruned}})$
34: **end procedure**

GOP (first 30 frames) of the *aspen* FHD sequence when the latter is encoded with H.264/AVC with target bitrate set to 500kbps. For downscaling, we use nine scale factors $S = \{1, 5/4, 4/3, 3/2, 2, 5/2, 3, 4, 6\}$ with our trained multi-scale precoding network (described in Section III), while upscaling is performed using the bilinear filter. Fig. 5a shows the set of RD points obtained after precoding the GOP for each scale factor in S and encoding with H.264/AVC via VBV encoding with the CRF values of Table I, which is representative of real-world streaming presets [34], and both maximum bitrate and buffer size set to 500kbps. As shown in Fig. 5b, the RD points (and the corresponding precoding modes) that do not provide for a monotonically increasing rate-PSNR curve are eliminated from the RD plane. Since
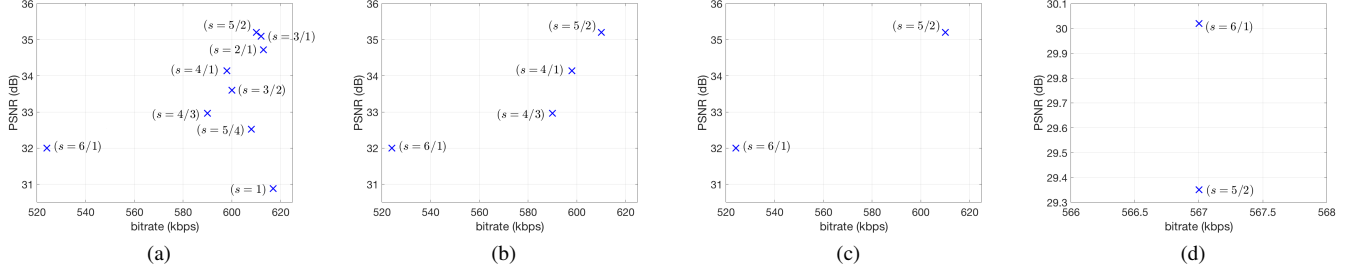
Fig. 5. Illustration of the operation of the proposed precoding mode selection algorithm during the encoding of the first 30 frames of the *aspen* FHD video sequence. (a) RD points corresponding to all encodings of the first GOP with H.264/AVC for all scale factors in $\boldsymbol{S}$. (b) Remaining RD points after the pruning of the points in (a) that do not provide for a monotonically increasing RD curve. (c) RD points remaining after the elimination of the points that do not lie on the convex hull of the RD points in (b). (d) Remapping of the RD points in (c) with CBR encoding at average bitrate after the pruning process.

there are more than two points remaining after this step, we next prune out the points that do not lie on the convex hull, which leaves us with two candidate precoding modes $s = 6$ and $s = 5/2$. We finally re-encode the GOP, precoded with the two modes, with CBR encoding at 567kbps, which is the average bitrate of the two points in Fig. 5c. This results in the two RD points shown in Fig. 5d. The selected precoding mode is $s^* = 6$, since it renders the highest PSNR value between the two points. Notice that the CBR encoding acts as an RD remapping and leads to both modes obtaining lower PSNR values than their PSNRs under VBV encoding. However, the absolute PSNR values are not relevant, since we are looking for the maximum PSNR under CBR; the final precoding mode is applied on the entire GOP and the encoding uses the VBV encoding mode preset. Finally, it is also of interest to notice that for this low-bitrate case, the $s = 1$ case (full resolution) is immediately shown to be suboptimal, and the mode selection is left to select from the higher-quality/higher-bitrate mode of $s = 5/2$ and the lower-quality/lower-bitrate mode of $s = 6$, with the latter prevailing when remapping the two points into CBR encoding at their average rate.

V. EXPERIMENTAL RESULTS

In this section, we evaluate our proposed adaptive video optimization framework in scenarios with the widest practical impact. We first compare the performance of the proposed multi-scale precoding network with the performance of standard linear downscaling filters for individual precoding modes. Then, we evaluate the entire adaptive video precoding framework by comparing it to standard video codecs via their FFmpeg open-source implementations, as well as to external highly-regarded video encoders that act as third-party anchors. The use of FFmpeg encoding libraries instead of the reference software libraries provided by the MPEG/ITU-T or AOMedia standardization bodies allows for the use of the same VBV model architecture for all tests and corresponds to a widely-used streaming-oriented scenario found in systems deployed around the world.

A. Content and Test Settings

The test content comprises 16 FHD (1920×1080) and 14 UHD (3840×2160) standard video sequences

TABLE I
CRF VALUES FOR LIBX264 AND LIBX265 AND SPEED FOR LIBVPX-VP9 UNDER OUR UTILIZED PRECODING MODES.

	Encoders		
s	libx264	libx265	libvpx-vp9
1	23	-	-
5/4	23	-	-
4/3	23	23	speed=1
3/2	23	23	speed=1
2	18	18	speed=1
5/2	18	18	speed=1
3	18	18	speed=1
4	18	18	speed=1
6	18	18	speed=1

in 8-bit YUV420 format from the XIPH collection[4], which have also been used in AOMedia standardization efforts. The FHD test content comprises the sequences *aspen*, *blue_sky*, *controlled_burn*, *rush_field_cuts*, *sunflower*, *rush_hour*, *old_town_cross*, *crowd_run*, *tractor*, *touchdown*, *riverbed*, *red_kayak*, *west_wind_easy*, *pedestrian_area*, *ducks_take_off*, *park_joy*, which have frame rates between 25fps and 50fps. The UHD sequences used in the tests are *Netflix_BarScene*, *Netflix_Boat*, *Netflix_BoxingPractice*, *Netflix_Crosswalk*, *Netfix_Dancers*, *Netflix_DinnerScene*, *Netflix_DrivingPOV*, *Netflix_FoodMarket*, *Netflix_FoodMarket2*, *Netflix_Narrator*, *Netflix_RitualDance*, *Netflix_RollerCoaster*, *Netflix_Tango*, *Netflix_TunnelFlag*, all at 60fps.

Performance is measured in terms of average PSNR and average VMAF, calculated with the tools made available by Netflix [13]. Average PSNR is the arithmetic mean of the PSNR values of all YUV channels of each frame. Similarly to PSNR, VMAF is measured per frame and the average VMAF is obtained by taking the arithmetic mean over all frames.

[4]https://media.xiph.org/video/derf/ The 4K (4096×2160) sequences used were cropped to the 3840×2160 (UHD) section of the central portion and were encoded to 8-bit YUV420 format (using x265 lossless compression) prior to encoding to produce UHD sequences. For the experiments of Section V-B, we used only the first 240 frames of these UHD sequences as well as the standard two-pass rate control settings in FFmpeg [46]. This corresponds to the typical configuration used in general-purpose video encoding. For the experiments of Section V-C, we used the single-pass VBV encoding settings for libx264/libx265/libvpx-vp9 that correspond to OTT streaming scenarios and can be found in FFmpeg documentation [47] and the downscaling factors and CRF values of Table I.

TABLE II
BD-RATE (ΔR) AND BD-PSNR (ΔP) RESULTS FOR REPRESENTATIVE DOWNSCALING FACTORS.

| | H.264/AVC | | | | H.265/HEVC | | | |
| | bicubic | | Lanczos | | bicubic | | Lanczos | |
s	ΔR	ΔP	ΔR	ΔP	ΔR	ΔP	ΔR	ΔP
5/2	-24.70%	0.61dB	-19.21%	0.45dB	-25.17%	0.55dB	-18.84%	0.39dB
2	-18.85%	0.56dB	-14.71%	0.42dB	-19.25%	0.52dB	-14.46%	0.37dB
3/2	-17.11%	0.45dB	-11.75%	0.31dB	-13.18%	0.32dB	-8.26%	0.20dB

TABLE III
BD-RATE (ΔR) AND BD-VMAF (ΔV) FOR REPRESENTATIVE DOWNSCALING FACTORS.

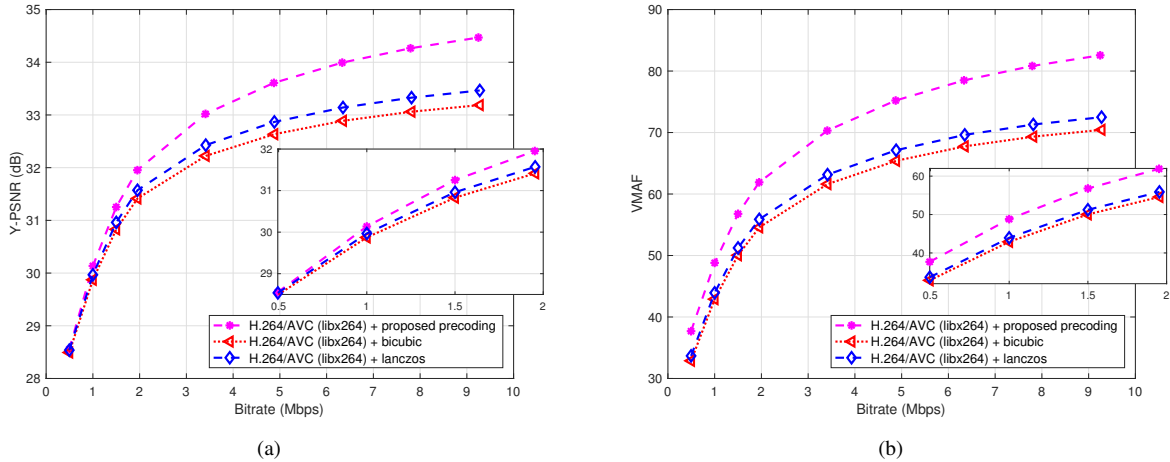| | H.264/AVC | | | | H.265/HEVC | | | |
| | bicubic | | Lanczos | | bicubic | | Lanczos | |
s	ΔR	ΔV	ΔR	ΔV	ΔR	ΔV	ΔR	ΔV
5/2	-39.74%	7.86	-34.30%	6.49	-39.73%	7.03	-33.75%	5.74
2	-30.32%	5.81	-27.57%	5.18	-30.20%	5.12	-27.41%	4.57
3/2	-23.21%	3.43	-21.73%	3.18	-18.66%	2.61	-17.67%	2.46

Fig. 6. Rate-distortion curves in terms of (a) PSNR and (b) VMAF for FHD content encoded with H.264/AVC and scale factor $s = 5/2$.

While PSNR has been used for decades as a visual quality metric, VMAF is a relatively recent perceptual video quality metric adopted by the video streaming community, which has been shown to correlate very well with the human perception of video quality. It is a self-interpretable $[0, 100]$ visual quality scoring metric that uses a pretrained fusion approach to merge several state-of-the-art individual visual quality scores into a single metric. Both PSNR and VMAF are calculated on native resolution frames after decoding and upscaling with the bilinear filter that is supported by all video players and web browsers.

B. Evaluation of Precoding Modes

We first evaluate the performance of our proposed multi-scale precoding network against the bicubic and Lanczos filters, which are the two standard downscaling filters supported by all mainstream encoding libraries like FFmpeg. We focus on three indicative scale factors on FHD content, opting

for a very common scenario of H.264/AVC encoding under its FFmpeg libx264 implementation. Specifically, we use the "medium" preset, two pass rate control mode [46], GOP=30, and bitrate range of $0.5 - 10$Mbps.

BD-rate [14] gains with respect to PSNR and VMAF are shown in Table II and Table III, respectively. Our precoding is shown to consistently outperform bicubic and Lanczos downscaling for all modes. For PSNR, its BD-rate gains ranged from 8% to 25%, while, for VMAF, rate reduction of 18%-40% is obtained. Indicative rate-distortion curves with respect to PSNR and VMAF for $s = 5/2$ scaling factor are presented in Fig. 6a and Fig. 6b, showing that the proposed precoding network consistently outperforms conventional downscaling filters. While the gain increases at higher bitrates, substantial gain is observed in the low bitrate region as well. Specifically, for PSNR, the BD-rate and BD-PSNR gains over bicubic (Lanczos) downscaling for the 0.5-2Mbps rate region (zoomed part of the curve in Fig. 6a) are 10.54% (6.9%) and 0.27dB (0.18dB), respectively. For VMAF, the BD-rate and BD-

Fig. 7. Two segments of frame 25 of the *crowd_run* FHD sequence encoded at 5000Kbps with the settings corresponding to Fig. 6. The precoded stream preserves the lettering and overall shapes significantly better than Lanczos and bicubic downscaling. Best viewed under magnification.
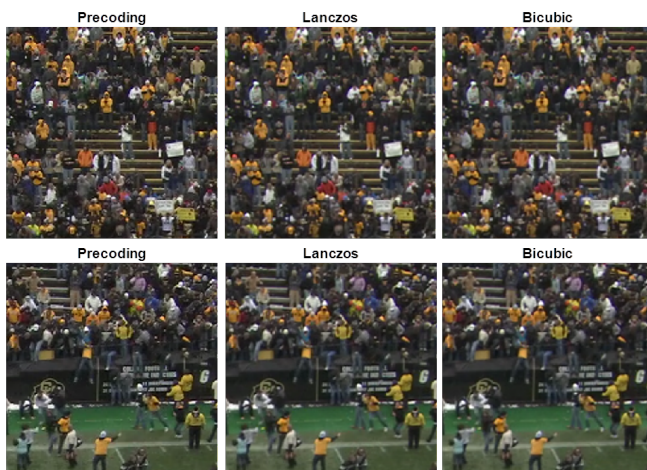

Fig. 8. Two segments of frame 77 of *rush_field_cuts* FHD sequence encoded at 5000Kbps with the settings of Fig. 6. The precoded stream preserves the geometric structures significantly better than Lanczos and bicubic downscaling. Best viewed under magnification.

VMAF gains over bicubic (Lanczos) downscaling are 29.29% (24.74%) and 5.94 (4.91), respectively, for the same low bitrate region. Example segments of a frame encoded at 5000kbps with the proposed precoding, and the Lanczos and bicubic downscaling are shown in Fig. 7 and Fig. 8. The improvement in visual fidelity demonstrated in the figures is also captured by the (approx.) 10-point average VMAF difference shown at the 5Mbps point of Fig. 6b. Several of the FHD video sequences encoded with a variation of the proposed precoding and H.265/HEVC (and the corresponding H.265/HEVC encoded results) are also available for visual inspection at www.isize.co/portfolio/demo. They can be played with any player, since our proposal does not require any change at the streaming or client side.

TABLE IV
AVERAGE BD-RATE (ΔR) AND BD-PSNR (ΔP) FOR 16 FHD TEST SEQUENCES AND SETTINGS DESCRIBED IN SECTION V-C. THE PROPOSED PRECODING, iSIZE, IN CONJUNCTION WITH H.264/AVC, H.265/HEVC AND VP9 IS EVALUATED AGAINST STANDALONE H.264/AVC, H.265/HEVC AND VP9 ENCODINGS AS WELL AS COMMERCIAL SOLUTIONS INCLUDING AWS MEDIACONVERT (AVC AND HEVC) AND AWS ELASTIC TRANSCODER (VP9).

	AVC + iSIZE		HEVC + iSIZE		VP9 + iSIZE	
	ΔR	ΔP	ΔR	ΔP	ΔR	ΔP
AVC	-14.80%	0.72dB	–	–	–	–
HEVC	–	–	-8.09%	0.27dB	–	–
VP9	–	–	–	–	-30.70%	1.21dB
AWS	-18.23%	0.86dB	–	–	-19.62%	1.17dB

TABLE V
AVERAGE BD-RATE (ΔR) AND BD-VMAF (ΔV) FOR 16 FHD TEST SEQUENCES AND SETTINGS DESCRIBED IN SECTION V-C. THE PROPOSED PRECODING, iSIZE, IN CONJUNCTION WITH H.264/AVC, H.265/HEVC AND VP9 IS EVALUATED AGAINST STANDALONE H.264/AVC, H.265/HEVC AND VP9 ENCODINGS AS WELL AS COMMERCIAL SOLUTIONS INCLUDING AWS MEDIACONVERT (AVC AND HEVC) AND AWS ELASTIC TRANSCODER (VP9).

	AVC + iSIZE		HEVC + iSIZE		VP9 + iSIZE	
	ΔR	ΔV	ΔR	ΔV	ΔR	ΔV
AVC	-26.29%	8.47	–	–	–	–
HEVC	–	–	-15.57%	3.03	–	–
VP9	–	–	–	–	-25.81%	6.07
AWS	-41.60%	12.18	–	–	-19.52%	6.75

C. Evaluation of Adaptive Precoding for Video-on-Demand Encoding

Since precoding can be applied to any codec and any video resolution, there is a virtually unlimited range of tests that can be carried out to assess its performance on multitude scenarios of interest. Here, we focus on test conditions appropriate for highly-optimized video-on-demand (VOD) encoding systems that are widely deployed today for video delivery. Our evaluation focuses on average bitrate-PSNR and birate-VMAF curves for our test FHD and UHD sequences and we present the results for: H.264/AVC in Fig. 9 and Fig. 10; H.265/HEVC in Fig. 11 and Fig. 12; and VP9 in Fig. 13 and Fig. 14. The corresponding BD-rate gains of our approach, which we term "iSize", in conjunction with each of these encoders vs. the corresponding encoder implementations, averaged over all test sequences, are presented in Tables IV-VII. For the proposed precoding method, we use footprinting with speed-up factor 5, i.e., only every 5th frame is processed during the selection of the best precoding mode, and the same encoding configuration is used as for the corresponding baseline encoder.

Regarding H.264/AVC and H.265/HEVC, we use the highly-optimized "slower" preset and VBV encoding for libx264/libx265, with GOP=90, and the widely-used crf=23 configuration for VBV (see footnote 4 for further details). For our approach, we employed (per codec) the precoding modes and crf values shown in Table I. To illustrate that our gains are achieved over a commercially competitive VOD encoding setup, for H.264/AVC we also include results

TABLE VI
AVERAGE BD-RATE (ΔR) AND BD-PSNR (ΔP) FOR 14 UHD TEST
SEQUENCES AND SETTINGS DESCRIBED IN SECTION V-C. THE PROPOSED
PRECODING, ISIZE, IN CONJUNCTION WITH H.264/AVC, H.265/HEVC
AND VP9 IS EVALUATED AGAINST STANDALONE H.264/AVC,
H.265/HEVC AND VP9 ENCODINGS AS WELL AS COMMERCIAL
SOLUTIONS INCLUDING AWS MEDIACONVERT (AVC AND HEVC) AND
AWS ELASTIC TRANSCODER (VP9).

	AVC + iSIZE		HEVC + iSIZE		VP9 + iSIZE	
	ΔR	ΔP	ΔR	ΔP	ΔR	ΔP
AVC	-52.30%	4.17dB	–	–	–	–
HEVC	–	–	-17.76%	0.58dB	–	–
VP9	–	–	–	–	-48.82%	2.03dB
AWS	-47.25%	3.77dB	–	–	-36.50%	2.02dB

TABLE VII
AVERAGE BD-RATE (ΔR) AND BD-VMAF (ΔV) FOR 14 UHD TEST
SEQUENCES AND SETTINGS DESCRIBED IN SECTION V-C. THE PROPOSED
PRECODING, ISIZE, IN CONJUNCTION WITH H.264/AVC, H.265/HEVC
AND VP9 IS EVALUATED AGAINST STANDALONE H.264/AVC,
H.265/HEVC AND VP9 ENCODINGS AS WELL AS COMMERCIAL
SOLUTIONS INCLUDING AWS MEDIACONVERT (AVC AND HEVC) AND
AWS ELASTIC TRANSCODER (VP9).

	AVC + iSIZE		HEVC + iSIZE		VP9 + iSIZE	
	ΔR	ΔV	ΔR	ΔV	ΔR	ΔV
AVC	-46.58%	15.52	–	–	–	–
HEVC	–	–	-19.68%	4.27	–	–
VP9	–	–	–	–	-33.32%	5.82
AWS	-68.70%	35.11	–	–	-67.77%	16.65

with the high-performing AWS MediaConvert encoder[5] using the MULTI_PASS_HQ H.264/AVC profile and its recently-announced high-performance QVBR mode with the default value of quality level 7. The results of Tables IV-VII show that, against the H.264/AVC libx264 implementation, the average rate saving of our approach for both FHD and UHD resolution under both metrics (PSNR and VMAF) is 35%; the corresponding saving of our approach against H.264/AVC AWS MediaConvert is 44%. For H.265/HEVC libx265, the average saving of our approach is 15%.

Regarding VP9, we employed VBV encoding with min-max rate (see more details at [49]), GOP=90 frames, maxrate=1.45×minrate, speed=1 for lower-resolution encoding (see Table I) and speed=2 for the full-resolution encoding anchor, since we only utilize downscaled versions with 6% to 64% of the video pixels of the original resolution. Additional bitrate reduction may be achievable by utilizing two-pass encoding in libvpx-vp9, but we opted not to use VBV encoding to make our comparison balanced with the VP9 implementation provided by the AWS Elastic Transcoder, which was used as our external benchmark for VP9. The settings of the Elastic Transcoder jobs were based on the built-in presets[6], which we customized to match the desired output video codec, resolution, bitrate, and GOP size, and we set

[5]AWS tools do not support H.265/HEVC, so no corresponding benchmark is presented for that encoder from implementations external to FFmpeg. However, libx265 is well recognized as a state-of-the-art implementation and is frequently used in encoding benchmarks [48].

[6]https://docs.aws.amazon.com/elastictranscoder/latest/developerguide/preset-settings.html

the framerate according to the input video framerate. Such customization is necessary because the built-in presets do not follow the input video parameters and they serve mainly as boilerplates. The results of Tables IV-VII show that, against the VP9 libvpx-vp9 implementation, the average rate saving of our approach for both FHD and UHD resolution under both metrics (PSNR and VMAF) is 35%; the corresponding saving of our approach against VP9 AWS Elastic Transcoder is 36%.

D. Further Comparisons and Discussion

1) Evaluation of adaptive precoding on HD (720p) content: To examine the performance of our approach for lower-resolution inputs, we carried out a smaller-scale evaluation of our proposed adaptive precoding method on four 1280×720 HD video sequences, namely *ducks_take_off*, *in_to_tree*, *old_town_cross* and *park_joy* from the XIPH collection. The average BD-rate gains of our approach versus the standalone H.264/AVC encoding for bitrates in the region of 0.5-2.5Mbps and same settings as in Section V-C were found to be 3.01% and 0.11dB for PSNR, and 2.27% and 0.68 for VMAF. These gains are modest in comparison to those obtained for FHD and UHD content. This indicates that our proposal is more suitable for high-resolution content.

2) Evaluation of adaptive precoding in conjunction with the current VVC Test Model (VTM): Beyond comparisons with existing standards, to show that our proposed deep video precoding can offer gains even over upcoming video coding standards, we evaluate our method in conjunction with the current VTM (version 6.2rc1). As the utilized VTM is very slow (2-10min per frame on single-core CPU execution), for this evaluation, we limited our tests to seven FHD sequences from the XIPH collection (*aspen, controlled_burn, old_town_cross, crowd_run, rush_field_cuts, touchdown, tractor*) and two bitrates: 1.8mbps and 3mbps. The utilized settings were as provided in the VTM software[7], with the default rate control enabled and set to the target bitrates and IntraPeriod is set to 64 frames. The results are summarized in Table VIII. We can see that rate reduction of 8%-9% is offered at slight increase of PSNR and VMAF. This first evaluation illustrates that our proposal can achieve gains even against the current VTM model of the Joint Video Experts Team. We would also like to note that the VVC standard is still under development and it is expected that these results will change depending on the features that will be included in the final version of the standard.

3) Evaluation against other CNN-based downscaling frameworks: To compare the core CNN designs of our precoding framework against recent work on CNN-based downscaling, we investigate the performance of our proposed precoding network under fixed downscaling ratio of $s = 2$ followed by bilinear upscaling on four standard test image datasets: Set5, Set14, BSDS100 and Urban100. Our benchmarks comprise bicubic and CNN-based [15] downscaling coupled with bicubic and Lanczos upscaling filters that are more complex than our bilinear upscaling. As summarized in Table IX, with the exception of Li *et al*. [15] on BSDS100 followed by Lanczos
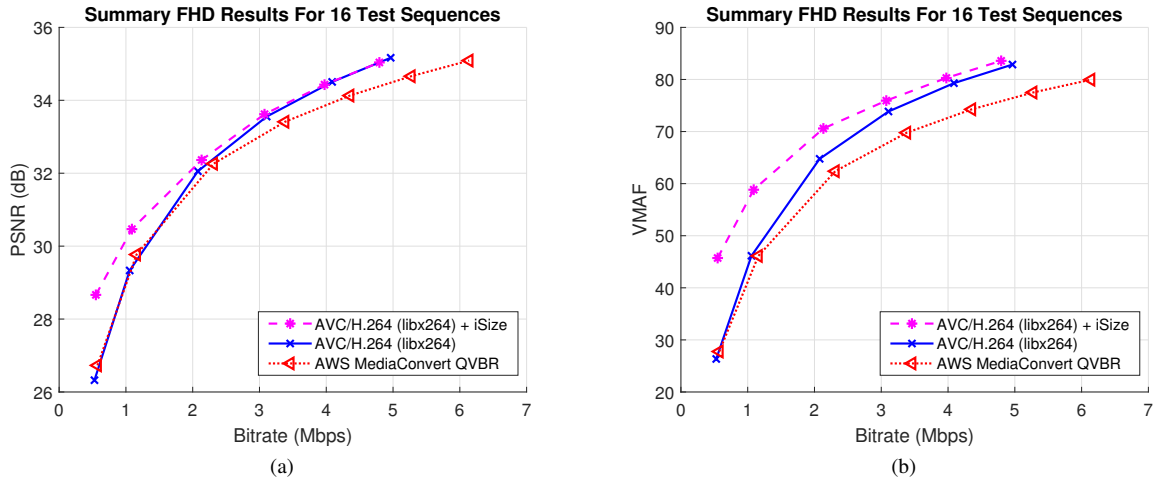
[7]https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM

Fig. 9. Performance comparison of H.264/AVC encoding with proposed adaptive precoding versus standalone H.264/AVC encoding and AWS MediaConvert H.264/AVC encoder (QVBR mode) on FHD content: (a) PSNR and (b) VMAF.
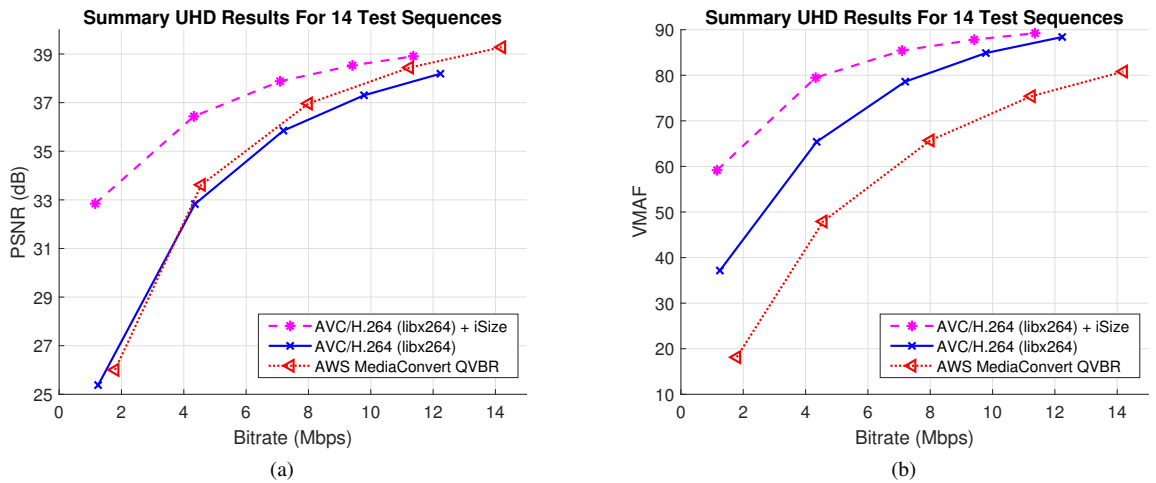

Fig. 10. Performance comparison of H.264/AVC encoding with proposed adaptive precoding versus standalone H.264/AVC encoding and AWS MediaConvert H.264/AVC encoder (QVBR mode) on UHD content: (a) PSNR and (b) VMAF.
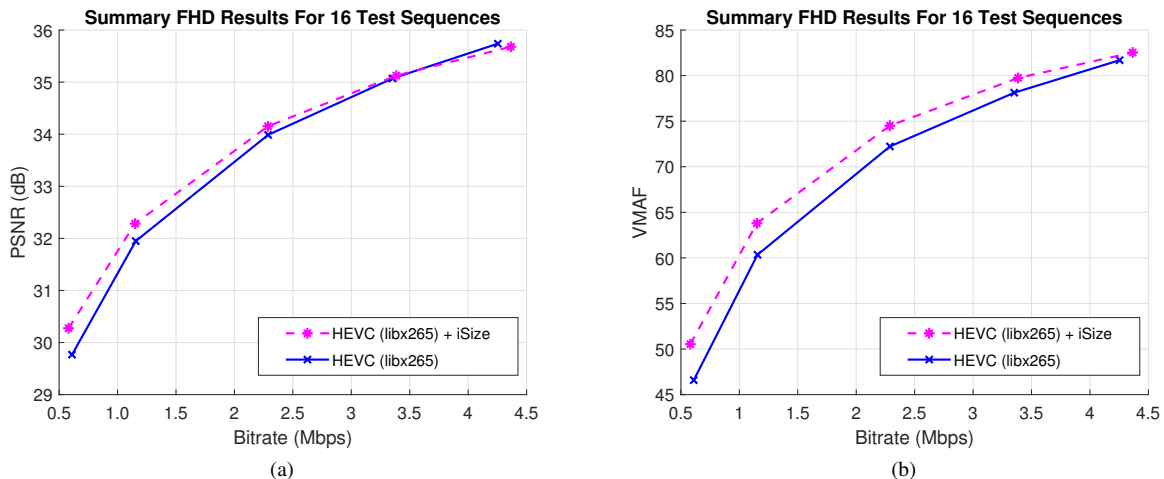

Fig. 11. Performance comparison of H.265/HEVC encoding with proposed adaptive precoding versus standalone H.265/HEVC encoding on FHD content: (a) PSNR and (b) VMAF.
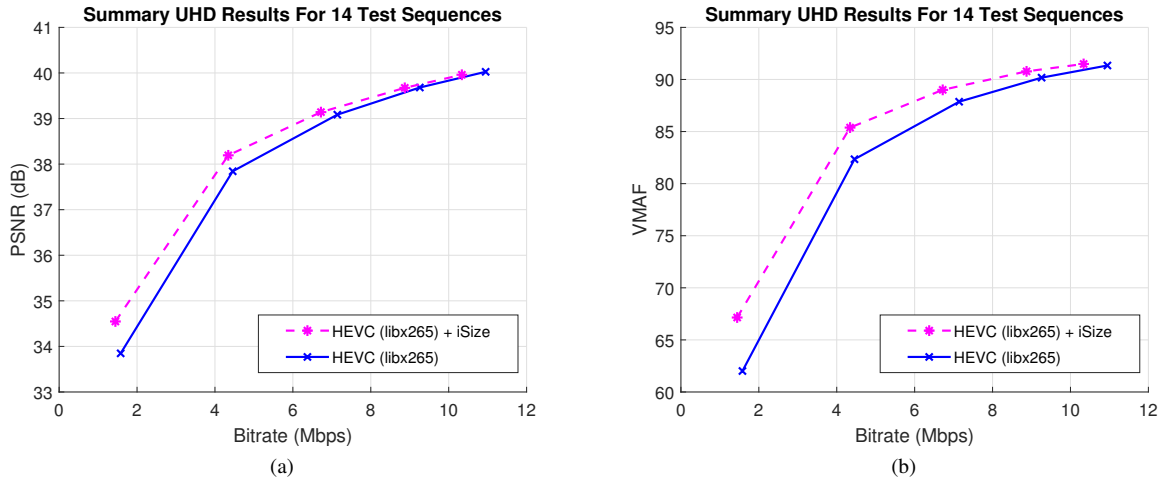
Fig. 12. Performance comparison of H.265/HEVC encoding with proposed adaptive precoding versus standalone H.265/HEVC encoding on UHD content: (a) PSNR and (b) VMAF.
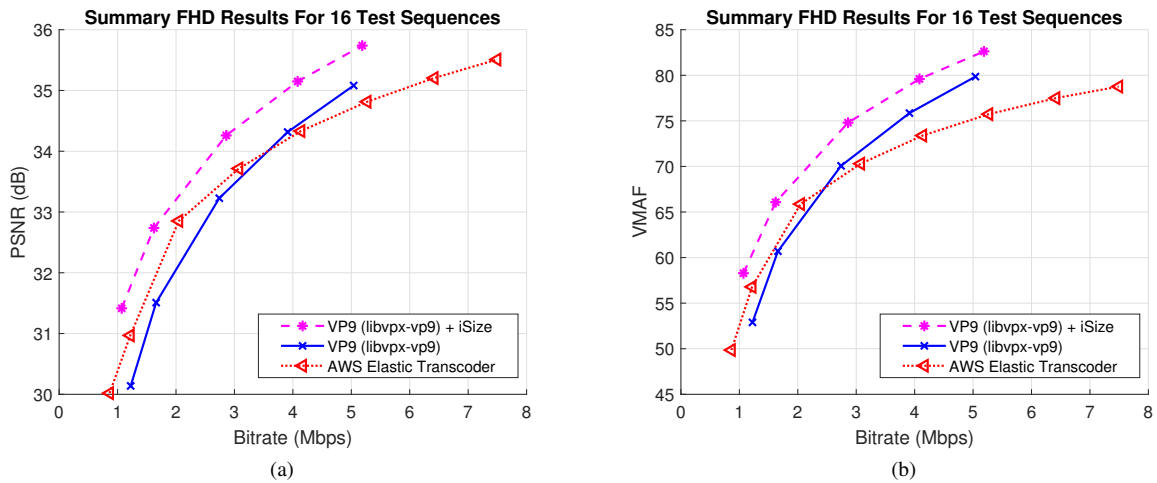

Fig. 13. Performance comparison of VP9 encoding with proposed adaptive precoding versus standalone VP9 encoding and AWS Elastic Transcoder VP9 encoder on FHD content: (a) PSNR and (b) VMAF
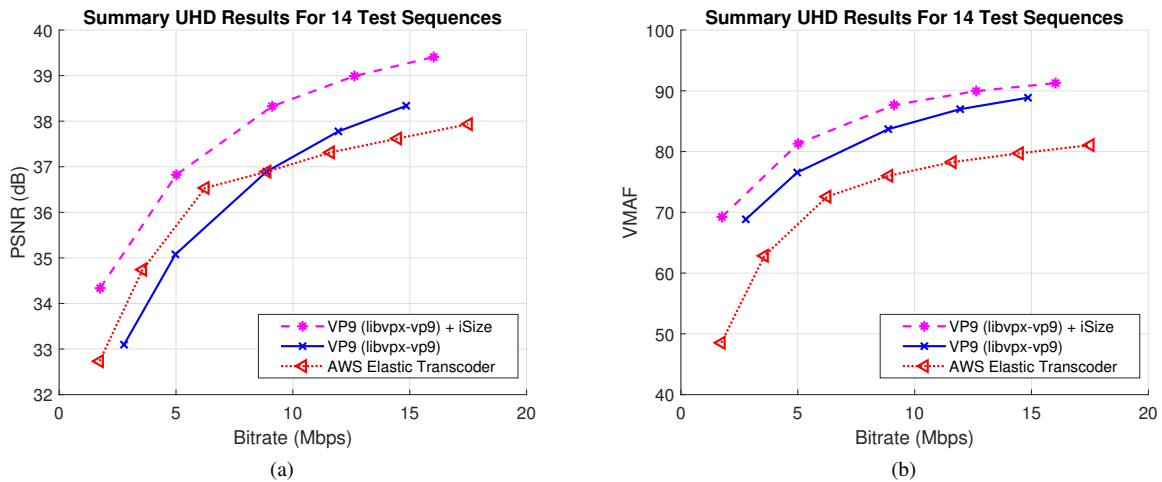

Fig. 14. Performance comparison of VP9 encoding with proposed adaptive precoding versus standalone VP9 encoding and AWS Elastic Transcoder VP9 encoder on UHD content: (a) PSNR and (b) VMAF.

TABLE VIII
AVERAGE PSNR, VMAF AND OBTAINED BITRATE FOR VVC ENCODING
(VTM V.6.2RC1) WITH PROPOSED PRECODING VERSUS STANDALONE
VVC ENCODING ON FHD CONTENT.

	VVC+iSize		VVC	
PSNR	27.22dB	28.18dB	27.15dB	28.04dB
VMAF	94.06	96.57	91.48	95.72
Bitrate (Mbps)	1.636	2.748	1.796	2.988

upscaling, our approach outperforms all reference methods on all datasets, and achieves this result with the very lightweight bilinear upscaling at the client side. In addition, our framework has significantly lower complexity than Li *et al.* [15], as for an 1920×1080 input frame and $s = 2$, our precoding network requires only 3.38G MACs and 5.5K parameters over all scales compared to 153G MACs and 30.6K parameters required by Li *et al.* [15].

4) Impact of edge preservation loss: The purpose of the edge preservation loss in Eq. (2) is to ensure structural preservation for the non-integer downscaling ratios that utilize bilinear downscaling instead of a stride. We explore the impact of the edge preservation loss by computing average PSNR and SSIM over the DIV2K validation set for $\lambda \in \{0, 0.5, 2, 5\}$ and indicative scaling factors. The results are reported in Table X. Without the edge preservation loss ($\lambda = 0$), the average PSNR and SSIM for scaling factor $s = 3/2$ that utilizes bilinear downscaling are 35.86dB and 0.957, respectively. Increasing lambda to 0.5, the PSNR and SSIM increase to 36.6dB and 0.962, respectively. A similar increase is exhibited for scaling factor $s = 4/3$, where PSNR increases from 37.96dB to 38.23dB. However, we notice that as the weight increases, the metrics saturate and there can be a detrimental effect on integer ratios, such as $s = 4$, where the mean absolute error (MAE) and feed-through from higher scaling factors is sufficient in ensuring fidelity to the input frame. Therefore, in order to ensure a balance in performance over all scaling factors, we set $\lambda = 0.5$ in all our results.

E. Runtime Analysis for Cloud-based VOD Encoding

Since VOD encoding configurations are typically deployed over a cloud implementation, it is of interest to benchmark the complexity of encoding with our deep video precoding modes versus the corresponding plain encoder. Table XI presents benchmarks for typical precoding scales on an AWS t2.2xlarge instance with each precoding and encoding running in two threads on two of the Intel Xeon 3.3GHz CPUs. The results correspond to the average processing of each GOP (comprising 90 frames and excluding I/O). By comparing the standard encoding time per video coding standard (scale 1) with our precoding and encoding time for the remaining scales, we see that, as scale increases, the encoding time reduces by up to a factor of five, while precoding time remains quasi-constant. The rate savings versus downscaling by these factors using bicubic and Lanczos filters are shown in Table II and Table III (for FHD content). These results indicate that, especially for the case of complex encoding standards like H.265/HEVC and VP9 that require long encoding times for high-quality

VOD streaming systems, the combination of precoding with the appropriate ratio may allow for a more efficient realization on a cloud platform with substantial reduction in rate (or improvement in quality) than using linear filters. In this context, precoding effectively acts as a data-driven pre-encoding compaction mechanism in the pixel domain, which allows for accelerated encoding, with the client linearly upscaling to the full resolution and producing high quality video.

VI. CONCLUSION

We propose the concept of deep video precoding based on convolutional neural networks, with the current focus being on downscaling-based compaction under DASH/HLS adaptation. A key aspect of our approach is that it remains backward compatible to existing systems and does not require any change for the streaming, decoder, and display components of a VOD solution. Given that our approach does not alter the encoding process, it offers an additional optimization dimension going beyond content-adaptive encoding and codec parameter optimization. Indeed, experiments show that it brings benefits on top of such well-known optimizations: under high-performing two-pass and VBV-based FHD and UHD video encoding, our precoding offers 8%-52% bitrate saving versus leading AVC/H.264, HEVC and VP9 encoders, with lower gains offered for HD (720p) content. An early-stage evaluation against the VVC Test Model v6.2rc1 showed that our approach may also be beneficial for advanced encoding frameworks currently under consideration by video coding standardization bodies. In addition, a comparison against a state-of-the-art CNN-based downscaling framework and bicubic or Lanczos upscaling showed that our proposal offers a better downscaler even when the less-complex bilinear filter is used for upscaling at the client side. The compaction features of our solution ensure that, not only bitrate is saved, but also that video encoding complexity reduction can be achieved, especially for HEVC and VP9 VOD encoding. Future work will consider how to extend the notion of precoding beyond adaptive streaming systems by learning to adaptively preprocess video inputs such that they are optimally recovered by current decoders under specified perceptual quality metrics.

REFERENCES

[1] I. Sodagar, "The MPEG-DASH standard for multimedia streaming over the internet," *IEEE multimedia*, vol. 18, no. 4, pp. 62–67, 2011.

[2] D. Weinberger, "Choosing the right video bitrate for streaming HLS and DASH," Feb 2015. [Online]. Available: https://bitmovin.com/video-bitrate-streaming-hls-dash/

[3] I. Katsavounidis and L. Guo, "Video codec comparison using the dynamic optimizer framework," in *Applications of Digital Image Processing XLI*, vol. 10752, International Society for Optics and Photonics. SPIE, 2018, pp. 266 – 281.

[4] M. Haris, G. Shakhnarovich, and N. Ukita, "Deep back-projection networks for super-resolution," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 1664–1673.

[5] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition Workshops (CVPRW)*, July 2017, pp. 1132–1140.

[6] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison, "Hidden technical debt in machine learning systems," in *Advances in neural information processing systems*, 2015, pp. 2503–2511.

TABLE IX
EVALUATION OF THE PROPOSED PRECODING NEURAL NETWORK COUPLED WITH BILINEAR UPSCALING FILTER AGAINST BICUBIC AND CNN-BASED DOWNSCALING METHODS PAIRED WITH DIFFERENT UPSCALING FILTERS IN TERMS OF RECONSTRUCTION QUALITY (PSNR) ON STANDARD IMAGE DATASETS FOR SCALING FACTOR $s = 2$.

	Bicubic $\downarrow$ Bilinear $\uparrow$	$CNN - CR^{Sep}$ [15] $\downarrow$ Bilinear $\uparrow$	Bicubic $\downarrow$ Bicubic $\uparrow$	$CNN - CR^{Sep}$ [15] $\downarrow$ Bicubic $\uparrow$	Bicubic $\downarrow$ Lanczos $\uparrow$	$CNN - CR^{Sep}$ [15] $\downarrow$ Lanczos $\uparrow$	Our precoding $\downarrow$ Bilinear $\uparrow$
Set5	32.26	33.55	33.67	33.36	33.72	33.37	**34.81**
Set14	28.97	30.27	30.01	30.30	30.05	30.32	**30.81**
BSDS100	28.70	29.80	29.57	29.86	29.61	**29.87**	29.67
Urban100	25.60	26.99	26.56	27.09	26.60	27.11	**27.65**
Average	27.38	28.63	28.32	28.70	28.36	28.72	**28.94**

TABLE X
EFFECT OF THE EDGE PRESERVATION LOSS ON PSNR/SSIM FOR INDICATIVE SCALING FACTORS ON THE DIV2K VALIDATION DATASET.

| | $\lambda = 0$ | | $\lambda = 0.5$ | | $\lambda = 2$ | | $\lambda = 5$ | |
s	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
4/3	37.96dB	0.97247	38.23dB	0.97384	38.21dB	0.97367	38.13dB	0.97331
3/2	35.86dB	0.95689	36.61dB	0.96223	36.61dB	0.96216	36.60dB	0.96211
3	29.88dB	0.84640	29.82dB	0.84634	29.77dB	0.84569	29.74dB	0.84548
4	28.47dB	0.79431	28.38	0.79346	28.28dB	0.79204	28.19dB	0.79088

TABLE XI
AVERAGE RUNTIME (MSEC PER GOP ON CPU) FOR REPRESENTATIVE DOWNSCALING FACTORS AND MULTIPLE ENCODING BITRATES.

| | Precoding | | AVC | | HEVC | | VP9 | |
s	FHD	UHD	FHD	UHD	FHD	UHD	FHD	UHD
1	0	0	1451	4928	11002	54016	3072	594400
3/2	671	4067	745	1982	7634	32565	2107	318445
2	775	4468	529	1854	7476	31786	1763	221522
5/2	703	4408	358	1008	6292	14692	1015	156259

[7] A. Wiesel, Y. C. Eldar, and S. Shamai, "Linear precoding via conic optimization for fixed MIMO receivers," *IEEE Trans. on Signal Processing*, vol. 54, no. 1, pp. 161–176, Jan 2006.

[8] C. Weidmann and M. Vetterli, "Rate distortion behavior of sparse sources," *IEEE Transactions on information theory*, vol. 58, no. 8, pp. 4969–4992, 2012.

[9] Y. Tsaig, M. Elad, P. Milanfar, and G. H. Golub, "Variable projection for near-optimal filtering in low bit-rate block coders," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 1, pp. 154–160, 2005.

[10] J. Kopf, A. Shamir, and P. Peers, "Content-adaptive image downscaling," *ACM Trans. on Graphics (TOG)*, vol. 32, no. 6, p. 173, 2013.

[11] A. C. Öztireli and M. Gross, "Perceptually based downscaling of images," *ACM Trans. on Graphics (TOG)*, vol. 34, no. 4, p. 77, 2015.

[12] C. G. Bampis, Z. Li, I. Katsavounidis, T.-Y. Huang, C. Ekanadham, and A. C. Bovik, "Towards perceptually optimized end-to-end adaptive video streaming," *arXiv preprint arXiv:1808.03898*, 2018.

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

[14] G. Bjontegaard, "Calculation of average psnr differences between rd-curves," *VCEG-M33*, 2001.

[15] Y. Li, D. Liu, H. Li, L. Li, Z. Li, and F. Wu, "Learning a convolutional neural network for image compact-resolution," *IEEE Trans. on Image Processing*, vol. 28, no. 3, pp. 1092–1107, March 2019.

[16] G. Georgis, G. Lentaris, and D. Reisis, "Reduced complexity superresolution for low-bitrate video compression," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 26, no. 2, pp. 332–345, 2016.

[17] J. Kim, J. Kwon Lee, and K. Mu Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 1646–1654.

[18] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee, "Enhanced deep residual networks for single image super-resolution," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 136–144.

[19] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *Proc. of the European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 391–407.

[20] J. Kim, J. Kwon Lee, and K. Mu Lee, "Deeply-recursive convolutional network for image super-resolution," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 1637–1645.

[21] M. Afonso, F. Zhang, and D. R. Bull, "Video compression based on spatio-temporal resolution adaptation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 1, pp. 275–280, 2018.

[22] Y. Li, D. Liu, H. Li, L. Li, F. Wu, H. Zhang, and H. Yang, "Convolutional neural network-based block up-sampling for intra frame coding," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 28, no. 9, pp. 2316–2330, Sep. 2018.

[23] J. Lin, D. Liu, H. Yang, H. Li, and F. Wu, "Convolutional neural network-based block up-sampling for HEVC," *IEEE Trans. on Circuits and Systems for Video Technology*, 2018.

[24] K. Liu, D. Liu, H. Li, and F. Wu, "Convolutional neural network-based residue super-resolution for video coding," in *Proc. of IEEE Visual Communications and Image Processing (VCIP)*, Dec 2018.

[25] D. Liu, Y. Li, J. Lin, H. Li, and F. Wu, "Deep learning-based video coding: A review and a case study," *arXiv preprint arXiv:1904.12462*, 2018.

[26] L. Theis, W. Shi, A. Cunnigham, and F. Huszár, "Lossy image compression with compressive autoencoders," in *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2017.

[27] O. Rippel and L. Bourdev, "Real-time adaptive image compression," in *Proc. Int. Conf. on Machine Learning (ICML)*, vol. 70, Aug. 2017, pp. 2922–2930.

[28] A. Shocher, N. Cohen, and M. Irani, ""zero-shot" super-resolution using deep internal learning," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3118–3126.

[29] N. Weber, M. Waechter, S. C. Amend, S. Guthe, and M. Goesele, "Rapid,

detail-preserving image downscaling," *ACM Trans. on Graphics (TOG)*, vol. 35, no. 6, p. 205, 2016.

[30] X. Hou, J. Duan, and G. Qiu, "Deep feature consistent deep image transformations: Downscaling, decolorization and HDR tone mapping," *arXiv preprint arXiv:1707.09482*, 2017.

[31] H. Kim, M. Choi, B. Lim, and K. Mu Lee, "Task-aware image downscaling," in *Proc. of the European Conference on Computer Vision (ECCV)*, 2018, pp. 399–414.

[32] F. Jiang, W. Tao, S. Liu, J. Ren, X. Guo, and D. Zhao, "An end-to-end compression framework based on convolutional neural networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 10, pp. 3007–3018, Oct 2018.

[33] O. Rippel, S. Nair, C. Lew, S. Branson, A. G. Anderson, and L. Bourdev, "Learned video compression," *arXiv preprint arXiv:1811.06981*, 2018.

[34] A. Asan, I.-H. Mkwawa, L. Sun, W. Robitza, and A. C. Begen, "Optimum encoding approaches on video resolution changes: A comparative study," in *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2018, pp. 1003–1007.

[35] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295–307, 2016.

[36] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Deep Laplacian pyramid networks for fast and accurate super-resolution," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 624–632.

[37] Y. Tai, J. Yang, and X. Liu, "Image super-resolution via deep recursive residual network," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 3147–3155.

[38] Y. Tai, J. Yang, X. Liu, and C. Xu, "Memnet: A persistent memory network for image restoration," in *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*, 2017, pp. 4539–4547.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. of the European Conf. on Computer Vision (ECCV)*. Springer, 2016, pp. 630–645.

[40] ——, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*, ser. ICCV '15. Washington, DC, USA: IEEE Computer Society, 2015, pp. 1026–1034.

[41] ——, "Deep residual learning for image recognition," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[42] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. of the13th Int. Conf. on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[43] E. Agustsson and R. Timofte, "NTIRE 2017 challenge on single image super-resolution: Dataset and study," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 126–135.

[44] S. Ma, Wen Gao, and Yan Lu, "Rate-distortion analysis for H.264/AVC video coding and its application to rate control," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 15, no. 12, pp. 1533–1544, Dec 2005.

[45] A. Ortega and K. Ramchandran, "Rate-distortion methods for image and video compression," *IEEE Signal Processing Magazine*, vol. 15, no. 6, pp. 23–50, Nov 1998.

[46] "FFmpeg," https://trac.ffmpeg.org/wiki/Encode/H.264#twopass.

[47] "FFmpeg," https://trac.ffmpeg.org/wiki/Encode/H.264#AdditionalInformationTips.

[48] J. D. Cock, A. Mavlankar, A. Moorthy, and A. Aaron, "A large-scale video codec comparison of x264, x265 and libvpx for practical VOD applications," in *Proc. of SPIE Applications of Digital Image Processing XXXIX*, vol. 9971, 2016.

[49] "FFmpeg," https://trac.ffmpeg.org/wiki/Encode/VP9.