2019

DEEP-LEARNING BASED PRECODING TECHNIQUES FOR NEXT-GENERATION VIDEO COMPRESSION

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

iSIZE Ltd, London, UK, www.isize.co, info@isize.co

ABSTRACT

Several research 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 AVC/H.264, HEVC, VVC, Google VP9 and AOMedia AV1, as well as existing container and transport formats. Such compatibility is a crucial aspect, as the video content industry and hardware manufacturers are expected to remain committed to supporting these standards for the foreseeable future.

We propose deep neural networks as precoding components for current and future codec ecosystems. In our current deployments for DASH/HLS adaptive streaming, this comprises downscaling neural networks. Precoding via deep learning allows for full compatibility to current and future codec and transport standards while providing for significant savings. Our results with HD content show that 23%-43% rate reduction takes place under a range of state-of-the-art video codec implementations. The use of precoding can also lead to significant encoding complexity reduction, which is essential for the cloud deployment of complex encoders like AV1 and MPEG VVC. Therefore, beyond bitrate saving, deep-learning based precoding may reduce the required cloud resources for video transcoding and make cloud-based solutions competitive or superior to state-of-the-art captive deployments.

INTRODUCTION

In just a few short 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 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 HD/UHD content is still plagued by broadband infrastructure problems.

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

HTTP Adaptive Streaming

To get round this problem, OTT content providers now stream resolution and bitrate ladders, standardized as DASH and HLS (dynamic adaptive streaming over HTTP and HTTP live streaming), which allow for the client device to switch to a range of lower resolutions and bitrates when the connection speed does not suffice for the high-quality/full-resolution video bitstream. However, the impact on quality from the widely-used bicubic downscaling can be quite severe. In principle, this could be remedied by post-decoding learnable video upscaling solutions [1], which have already been shown to improve image quality in comparison to linear upscaling filters. However, their deployment requires very substantial changes in the decoding client device, which are usually too cumbersome and complex to make in practice. 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.

Introducing the Concept of Precoding for Video Communications

Precoding has been initially proposed for MIMO wireless communications as the means to preprocess the transmitted symbols and perform transmit diversity [2]. Precoding is similar to channel equalization, but the key difference is that one has to optimize the precoder with the operation of the utilized decoder. 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 video delivery. Our current focus is on HTTP adaptive video streaming, which is at the core of OTT video delivery. Precoding for video transport over HTTP is done by preprocessing the video signal prior to standard encoding, while allowing a standard video encoder and DASH/HLS compliant player to decode it without any modifications. As illustrated in Figure 1, the key principle is to optimize the operation of the precoder to minimize the distortion at the output of a DASH/HLS enabled player. This is achieved by optimally adjusting the utilized resolution according to: the bitrate, the utilized video codec and its specification, and the upscaling filter used by the DASH/HLS-compliant player. Concerning the latter, web browsers tend to use the bilinear filter[2] to ensure smooth playback in thin clients and mobile devices.

By utilizing deep learning as a tool for efficient precoding, a question at the core of our work is: *Can we reduce the distortion of standard DASH/HLS video playback via precoding with convolutional neural networks (CNNs)?* Following the schematic of Figure 1, we show that this is indeed possible per group of pictures (GOP) without any modification at the client (video decoder and player). Our experiments with standard test content from the XIPH repository and the x264/x265/libvpx-vp9 implementations of H.264/AVC, HEVC and VP9 encoders show that, in comparison to the use of linear downscaling filters, 14%-55% bitrate reduction is achieved for HD and UHD encoding over standard bitrates and configurations used in commercial deployments. An important effect of precoding is that video encoding and decoding 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

[2] Despite the abundance of 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.

factor. This is beneficial when considering cloud deployments of such encoders, especially in view of upcoming standards like AV1 and MPEG VVC that have substantially increased encoding complexity. Alternatively, such acceleration can be traded off for more complex encoding presets for lower-resolution inputs, which allows for further bitrate saving.

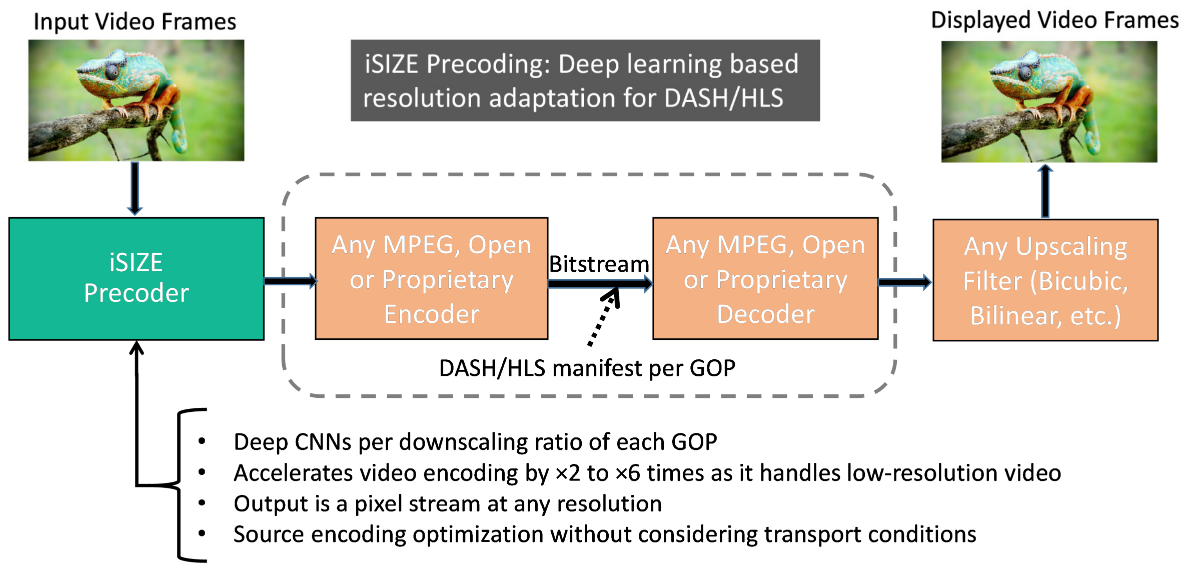
Input Video Frames
iSIZE Precoding: Deep learning based resolution adaptation for DASH/HLS
Displayed Video Frames
iSIZE Precoder
Any MPEG, Open or Proprietary Encoder
Bitstream
Any MPEG, Open or Proprietary Decoder
Any Upscaling Filter (Bicubic, Bilinear, etc.)
DASH/HLS manifest per GOP
• Deep CNNs per downscaling ratio of each GOP
• Accelerates video encoding by ×2 to ×6 times as it handles low-resolution video
• Output is a pixel stream at any resolution
• Source encoding optimization without considering transport conditions

Figure 1. Proposed deep-learning based precoding for DASH/HLS video streaming. Adaptation with convolutional neural networks (CNNs) occurs per group of pictures (GOP). The decoder side can optionally be adjusted to achieve further quality enhancement.

It is important to emphasize *the key difference of precoding with conventional DASH/HLS ladder optimization*: Precoding is a source encoding optimization approach and is carried out at the server-side prior to transmission in order to optimally select and derive the best downscaled representation per input GOP segment, bitrate and codec specification, *without considering transport conditions*. That is, once the DASH/HLS ladder of bitrates is produced by our approach (with the GOP resolution and downscaling CNN per bitrate selected automatically by our approach), adaptive bitrate streaming, stream caching and stream switching mechanisms that cope with bitrate and channel fluctuations can operate as usual. The key difference is that they receive, decode (and upscale) bespoke representations created by the precoder CNNs. Hence, our proposal for deep-learning based precoding is reducing the distortion of the DASH/HLS video player per bitrate by exploiting the fact that upscaling is supported by the player and precoding remains agnostic to the channel conditions during each individual video bitstream delivery.

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 combination of perceptual metrics, i.e., lowering the encoding bitrate for scenes that are deemed to be simple enough for a standard encoder to process with 50% (or less) bits. Such solutions can also be extended to other encoder parameters, and their essence is in the coupling of a visual quality profile to a pre-cooked encoder-specific tuning recipe.

Resolution adaptation in image and video coding has been explored by several authors. Recently, Katsavounidis *et al.* [3][4] 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 and encoding settings each group of pictures in the video. 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 VMAF [5]. While BD-rate [6] gains of 30% have been shown in experiments for H.264/AVC and VP9, the dynamic optimizer is extremely costly to deploy in an operational scenario and still uses non-learnable downscaling filters.

Overall, while such methods have shown the possibility of rate saving via image and video downscaling, they have not managed to outperform classical bicubic downscaling within the context of practical encoding. This has led most researchers 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. To this end, deep convolutional neural network (CNN) architectures have set the state-of-the-art, with recent deep CNNs like VDSR [1] and EDSR [7] achieving several dB higher PSNR in the luminance channel of standard image benchmarks for lossless image upscaling.

Inspired by these successes and the success of autoencoders for image compaction, Wave One recently proposed video encoding with deep neural networks [8] and demonstrated quality gains against a conventional video encoder without B frames, and focusing on very-high bitrate encoding (20mbps or higher for HD). While this is an important achievement, such solutions have not yet shown superior performance against modern video encoders when the latter utilize all their features (e.g., VBV encoding and their most advanced settings like AVC/H.264 libx264 “slower” preset). In addition, they require advanced GPU capabilities on both the client and the server side and do not support standards for video transport and decoding. Therefore, despite the advances that may be offered by all these methods in the future, they are not compatible with video coding standards and do not consider the stringent complexity constraints imposed on video streaming under DASH or HLS-compliant thin clients like tablets and mobile devices. Our work fills this gap by offering a deep-learning based solution that can operate in its entirety on the server side and does not require any change in the video transport, decoding and display side.

PRECODING NETWORKS

The proposed multi-scale precoding neural network comprises a series of precoding blocks, which progressively downscale high resolution (HR) frames over multiple scale factors corresponding to those of any designated DASH/HLS ladder and operate entirely at the encoder side prior to transmission. The resulting low resolution (LR) frames can then be encoded by the codec with lower complexity and higher coding efficiency, because we design the precoding CNNs to compact information in ways that the standard linear upscaler at the player side will be able to recover more efficiently than the case of a linear downscaler. Regarding encoding/decoding complexity saving, for downscaling by factor s, given $s > 1$, the number of pixels to encode is a factor of s^2 less than the original frame. The downscaled video can be optionally transcoded and streamed at a substantially lower bit-rate to a thin client (e.g., a mobile device). On the client side, the video is decoded and upscaled to the original (HR) resolution, with a simple linear upscaler, with the most common case being the bilinear filter. This is contrary to 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 upscaling [7] comprises over 40 million parameters and would therefore be impractical on the client side for 30-60 frame-per-second (fps) HD/UHD video.

In the following sections, we summarize the design methodology of the proposed multi-scale precoding networks, including the network architecture and online selection of best precoding mode for each GOP.

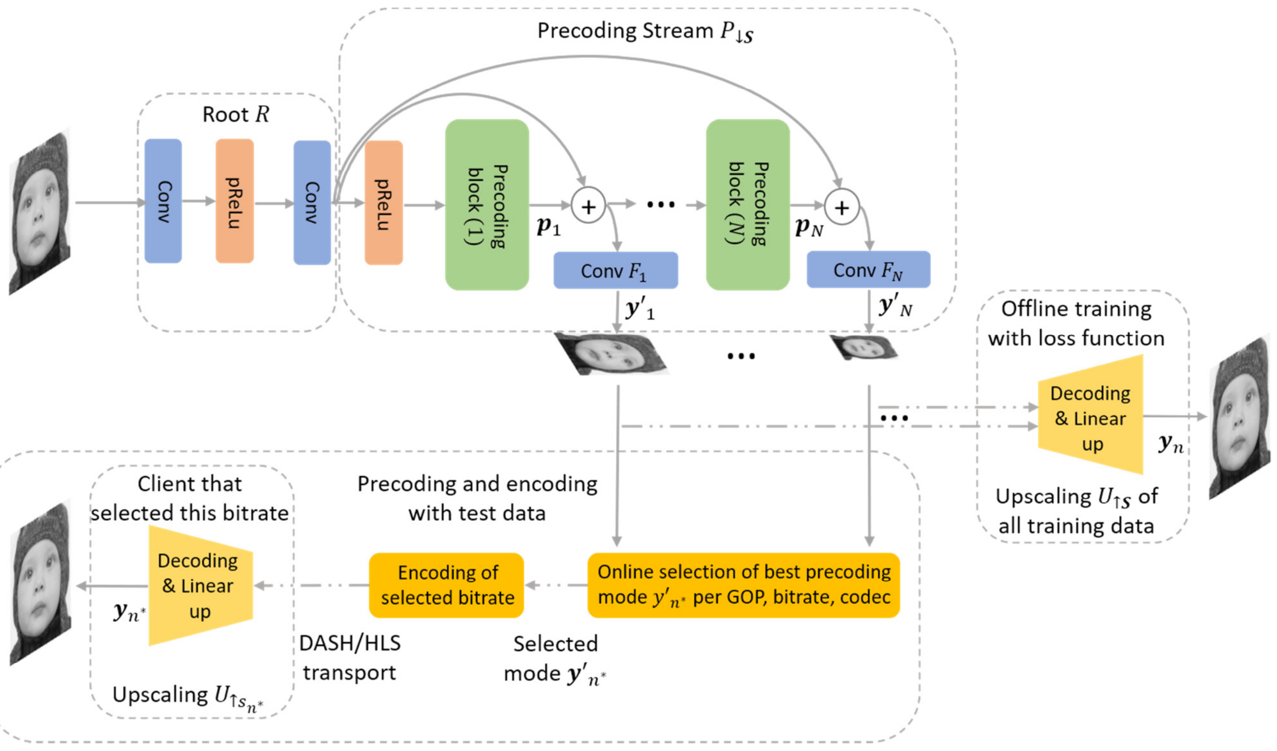
Precoding Stream $P_{\downarrow s}$
Root R
Conv
pReLu
Conv
pReLu
Precoding block (1)
p_1
Conv F_1
y'_1
Precoding block (N)
p_N
Conv F_N
y'_N
Offline training with loss function
Decoding & Linear up
y_n
Upscaling $U_{\uparrow s}$ of all training data
Client that selected this bitrate
Decoding & Linear up
y_{n^*}
Upscaling $U_{\uparrow s_{n^*}}$
Precoding and encoding with test data
Encoding of selected bitrate
Online selection of best precoding mode y'_{n^*} per GOP, bitrate, codec
DASH/HLS transport
Selected mode y'_{n^*}

Figure 2. Multi-scale precoding network for video downscaling prior to encoding, comprising a root mapping R, single precoding stream P and linear upsampling U. Each input frame of a video is downsampled by the precoding network at the server via N precoding blocks, corresponding to the N DASH/HLS adaptation points of the client, which are used during network training. At test deployment, the downscaled frames of the selected mode are passed to the codec for encoding at any chosen bitrate. During live streaming, the client device selects a bitrate per GOP and decodes and upsamples the downscaled video with simple linear upscaling, e.g., the bilinear filter.

Network Architecture

We visualize the precoding network in Figure 2 for progressive downscaling of individual luminance (Y) frames $\boldsymbol{x} \in \mathbb{R}^{H \times W}$ (where H and W are the height and width respectively) over multiple scales. Given that viewers are most sensitive to luma information, we intentionally process only the Y channel and not the chrominance (Cb, Cr) channels, in order to avoid unnecessary computation. Dong *et al.* [9] 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 can be downscaled and processed independently.

We first perform feature extraction (without downscaling) using a "root" mapping R, comprising two convolutional layers, in order to extract a set of high-dimensional feature maps $\boldsymbol{r}$ from the input $\boldsymbol{x}$. The extracted feature maps are then passed to the precoding stream. A precoding stream $P_{\downarrow \boldsymbol{S}}$ (or P for short-hand) comprises a sequence of N precoding blocks, which progressively downscale the input over a set of N scale factors, $\boldsymbol{S} = \{s_1, s_2, \dots, s_N\}$, where $s_1 < s_2 \dots < s_N$ and $\downarrow$ indicates downscaling. For N precoding blocks and N scale factors $\boldsymbol{S}$, we expect the embedding stream to output a set of N downscaled representations $\boldsymbol{Y}' = \{\boldsymbol{y}_1', \boldsymbol{y}_2', \dots, \boldsymbol{y}_N'\}$ of the input $\boldsymbol{x}$. The output activations in $\boldsymbol{Y}'$ are then clipped (rescaled) between the minimum and maximum pixel intensities and each representation can thus be used by the video codec as a downscaled low resolution (LR) frame. These frames can then be upscaled to the original resolution using linear upscaling $U_{\uparrow \boldsymbol{s}}$ on the client side, such as bilinear, lanczos or bicubic. We refer to the nth generated upscaled frame as $\boldsymbol{y}_n$.

Global Residual Learning

As illustrated in Figure 2, our precoding stream P utilizes a global residual learning strategy, where we use a skip connection and perform a pixel-wise summation between the root feature maps $\boldsymbol{r}$ (pre-activation function and after linear downscaling to the correct resolution) and the outputs of each precoding block. Similar global residual learning implementations have also been adopted by SR models [1][7]. In our case, our precoding stream effectively follows a pre-activation configuration 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.

Precoding Block

Our precoding blocks, which constitute the primary components of our network, are illustrated in Figure 3. The precoding block consists of alternating 3×3 and 1×1 convolutional layers, where each layer is followed by a parametric ReLu (pReLu) 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 nth precoding block is effectively responsible for downscaling by factor s_n / s_{n-1}. In order to maintain low complexity, it is important to downscale as early as possible. We therefore group all downsampling operations with the first convolutional layer in each precoding block. If s_n/s_{n-1} is divisible by 2, we simply use a stride of 2 in the first convolutional layer. If s_n/s_{n-1} is not divisible by 2, we use a stride of 1 in first convolutional layer and precede this with a linear bilinear/bicubic downscaling to the correct scale factor.

The precoding block must also be able to remove aliasing artifacts generated by downsampling with a stride: considering that the upscaling operations are only linear and therefore heavily constrained, the network is asymmetric and the precoding network cannot simply learn to invert the upscaling. A traditional linear anti-aliasing filter is a low pass filter that removes the high frequency components of the image, such that the image can be properly resolved. However, by removing the high frequencies from the image, this leads to blurring. Deblurring is an inverse problem, as there is typically limited information in the blurred image to uniquely determine a viable input. As such, we can respectively model the anti-aliasing and deblurring processes as a function composition of linear and non-linear mappings. As shown in Figure 3, this is done with a skip connection between linear and non-linear paths below, 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.

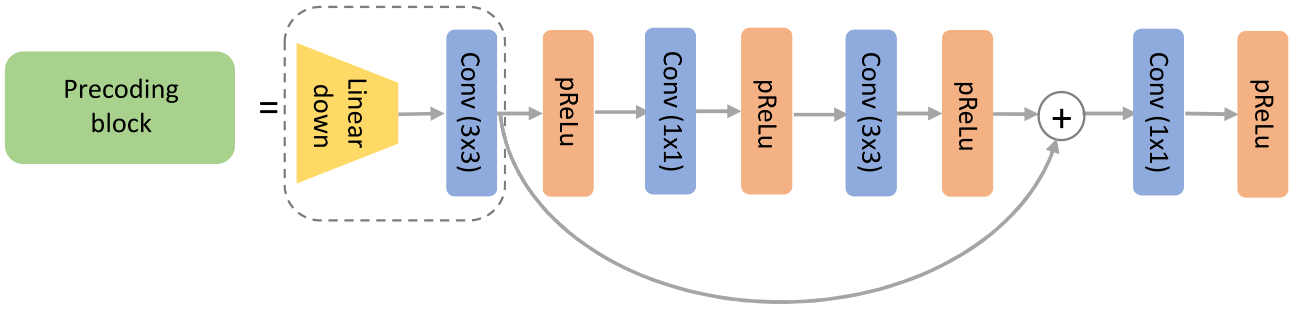

Figure 3. Block design for efficient precoding, comprised of a series of 3×3 and 1×1 convolutions. The first layer is grouped with a downscaling operation either via stride in the convolutional layer or with a preceding bicubic/bilinear downscaling. The linear mapping learned from the first layer (pre-activation function) is passed to the output of the second 3×3 conv layer (post-activation function) with a skip connection and pixel-wise summation.

Online Selection of Best Precoding Mode during GOP Encoding

Once the multiscale precoding architecture of Figure 2 has been trained on representative content, each GOP segment of the video is precoded into all possible scales, and an online selection process determines the best precoding mode to use. The key idea is to run through as many precoding modes as possible and evaluate a rate-distortion function that determines the rate-distortion characteristics of the video encoder if it would use this mode. Evaluating the rate-distortion characteristics determines the best mode per GOP, which is then used for the actual encoding. The decoder simply decodes the provided mode and upscales to the full resolution using the player's built-in upscaling filter. Given that resolution adaptation is already supported per DASH/HLS video segment, no modification is required in the entire bitstream, transport, DASH/HLS and client side in order to support our approach.

Multiple rate-distortion functions can be selected for the evaluation of the best precoding mode. For example, one can utilize operational rate-distortion models for H.264/AVC or

HEVC [10]. Instead, we perform selective encoding of a few frames in each GOP with a process we call “footprinting”. Once the rate-MSE[3] characteristics of these fast encodings are extracted from the utilized codec, we prune out all points that do not provide for a convex rate-distortion (RD) curve. From the remaining points, we select the point that corresponds to the lowest distortion when all points are re-encoded to their average bitrate with constant-bitrate encoding. This footprinting, pruning and remapping function to select the best point is quick to compute if only a few frames are used, and still remains codec agnostic, as it utilizes the codec to obtain the operational RD points needed.

Implementation and Training Details

In our proposed multi-scale precoding network, all kernels are initially set up using Xavier initialization [11]. We use parametric ReLu (pReLu) as the activation function, as indicated in Figure 2 and Figure 3, and zero padding to ensure that all layers are of the same size. Downscaling is only controlled by downsampling operations such as a stride. The root mapping R comprises a single 3×3 and 1×1 convolutional layer. We set the number of channels in all 1×1 and 3×3 convolutional layers to 4 and 8 respectively (excluding single channel convolutional layers F). In this way, 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 for a 1920×1080 input frame (including root mapping and all precoding streams).

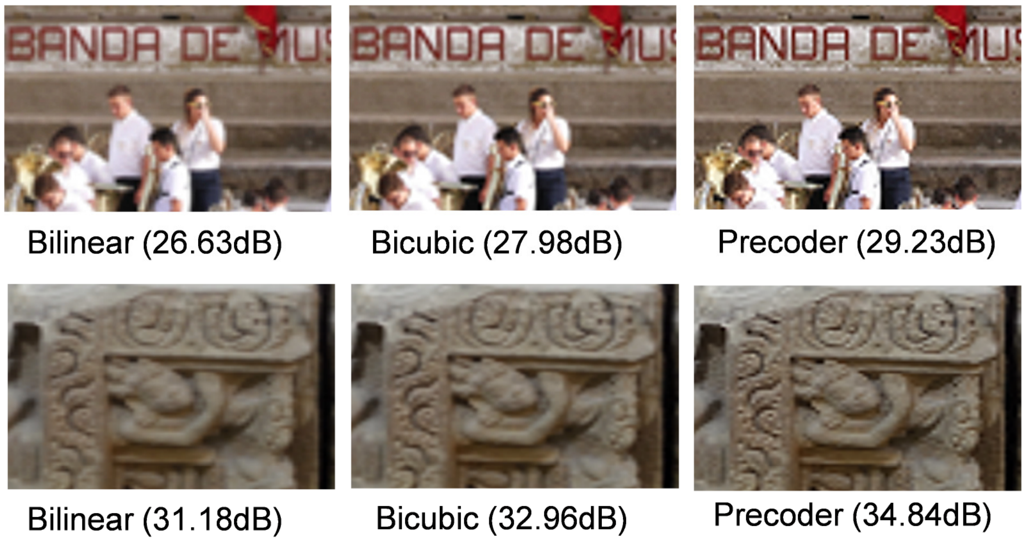

Figure 4. Subjective and objective comparison on examples from the DIV2K validation set.

We train the root module and all precoding streams end-to-end with linear upscaling (without any encoding) on images from the DIV2K [12] training set using a composite mean-absolute error (MAE) and a gradient-based loss function. This means that our

[3] Even though we could use other distortion measures, mean square error (MSE) is quick to compute and is automatically provided by all encoders.

training objective is not only to minimize MAE in a data-dependent manner, but also to take into account the structural aspects of the training images. All models were trained 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. Importantly, in order to ensure our implementation matches that of standard linear upscaling (e.g., FFmpeg or OpenCV), we wrote the all linear upscaling functions from scratch and did not use Tensorflow's in-built functions. Indicative results for portions of two validation images of DIV2K are given in Figure 4.

EVALUATION OF PRECODING MODES ON HD AND UHD VIDEO SEQUENCES

In order to assess our proposal for scenarios with the widest practical impact, we evaluate on standard scale factors for HD and UHD video coding within typical bitrate regimes and focus on the two MPEG standards in mainstream use today, i.e., H.264/AVC and HEVC under their FFmpeg libx264 and libx265 implementations and standard test content[4]. After standard bilinear upscaling with the libyuv that is supported by all players and web browsers, distortion is measured via PSNR and VMAF [5], with the latter being used by the video streaming community as a self-interpretable 0-100 visual quality metric that fuses several individual quality scores.

	AVC/H.264				**HEVC**			
	bicubic		**lanczos**		**bicubic**		**lanczos**	
Factor	**BD-rate**	**BD-PSNR**	**BD-rate**	**BD-PSNR**	**BD-rate**	**BD-PSNR**	**BD-rate**	**BD-PSNR**
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 1. BD-rate savings of our precoding for HD sequences for PSNR.

	AVC/H.264				**HEVC**			
	bicubic		**lanczos**		**bicubic**		**lanczos**	
Factor	**BD-rate**	**BD-VMAF**	**BD-rate**	**BD-VMAF**	**BD-rate**	**BD-VMAF**	**BD-rate**	**BD-VMAF**
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

Table 2. BD-rate savings of our precoding for HD sequences for VMAF.

[4] Setup: libx264/libx265 "medium" preset, GOP=30 frames and 0.5-10mbps bitrate range for HD, GOP=50 and 3-17mbps range for UHD, two-pass rate control: https://trac.ffmpeg.org/wiki/Encode/H.264#twopass; the test content from https://media.xiph.org/video/derf/ website was: 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. The UHD sequences used were (first 240 frames only, cropping to 3840×2160 of the central portion and lossless conversion to YUV420 format prior to encoding): Barscene, Boat, Crosswalk, Dancers, Dinnerscene, DrivingPOV, Foodmarket, Foodmarket2, Narrator.

Bjontegaard distortion-rate [6] gains are shown in Table 1-Table 4. Our precoding is shown to be particularly effectively for the case of VMAF, where rate reduction of 18%-40% and 35%-55% is obtained for HD sequences and UHD sequences, respectively. For the case of UHD sequences, the two modes shown sufficed for the coverage of the entire UHD bitrate region (2.5-10mbps), so mode 3/2 downscaling factor is omitted.

	AVC/H.264				**HEVC**			
	bicubic		**lanczos**		**bicubic**		**lanczos**	
Factor	**BD-rate**	**BD-PSNR**	**BD-rate**	**BD-PSNR**	**BD-rate**	**BD-PSNR**	**BD-rate**	**BD-PSNR**
5/2	-19.66%	0.26dB	-12.60%	0.17dB	-18.93%	0.25dB	-11.71%	0.15dB
2	-6.45%	0.11dB	-2.99%	0.05dB	-8.30%	0.12dB	-4.39%	0.07dB

Table 3. BD-rate savings of our precoding for UHD sequences for PSNR.

	AVC/H.264				**HEVC**			
	bicubic		**lanczos**		**bicubic**		**lanczos**	
Factor	**BD-rate**	**BD-VMAF**	**BD-rate**	**BD-VMAF**	**BD-rate**	**BD-VMAF**	**BD-rate**	**BD-VMAF**
5/2	-55.15%	5.87	-48.05%	4.98	-52.19%	5.41	-45.26%	4.57
2	-36.99%	4.34	-34.57%	4.02	-39.69%	3.93	-37.60%	3.69

Table 4. BD-rate savings of our precoding for UHD sequences for VMAF.

INDICATIVE EVALUATION OF ADAPTIVE PRECODING VERSUS CODEC RESULTS

Since precoding can be applied to any codec and any video input resolution, there is a virtually unlimited range of tests that can be carried out to assess its performance on multitude scenarios of interest. We present here two representative cases as illustrative examples of the boosting effect that deep-learning based precoding can have in existing codecs and transport ecosystems without breaking standard compliance.

The first one, given in the left part of Figure 5, relates to improving the performance of VP9 encoding for HD content over the entire range of quality-bitrates of commercial interest. To this end, we enabled multiple precoding modes (downscaling factors ranging from $s = 5/4$ to $s = 4$) and utilized the 16 HD sequences used in our previous tests in conjunction with libvpx-vp9 of FFmpeg[5]. To illustrate the competitiveness of the utilized encoding setup, we also provide the corresponding results of the AWS Elastric Transcoder. 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 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

[5] Setup: constant quality encoding with min-max rate (see https://trac.ffmpeg.org/wiki/Encode/VP9), GOP=90 frames, maxrate=1.45×minrate, speed=0 for lower-resolution encoding and speed=2 for full-resolution encoding, since we encode downscaled versions with 6% to 64% of the video pixels of the original resolution. Note that additional bitrate reduction can be achieved by utilizing VBV 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.

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

mainly as boilerplates. Our proposal is found to offer 23% and 30% rate reduction over libvpx-vp9 and the Elastic Transcoder, respectively.

The second scenario relates to improving the performance of an already highly-optimized H.264/AVC encoder for HD and UHD content and avoiding the switch to a more complex codec like HEVC or VP9. The right part of Figure 5 shows the results obtained with H.264/AVC[7] and the proposed precoding. To illustrate that our gains are achieved over a commercially competitive setup, we included results with AWS MediaConvert using the MULTI_PASS_HQ AVC/H.264 profile, which we customized to set the output video resolution and bitrate. For the QVBR mode of MediaConvert, we used the default value of quality level 7. For HD encoding, our proposal offers 28% and 43% rate reduction over libx264 and MediaConvert, respectively.

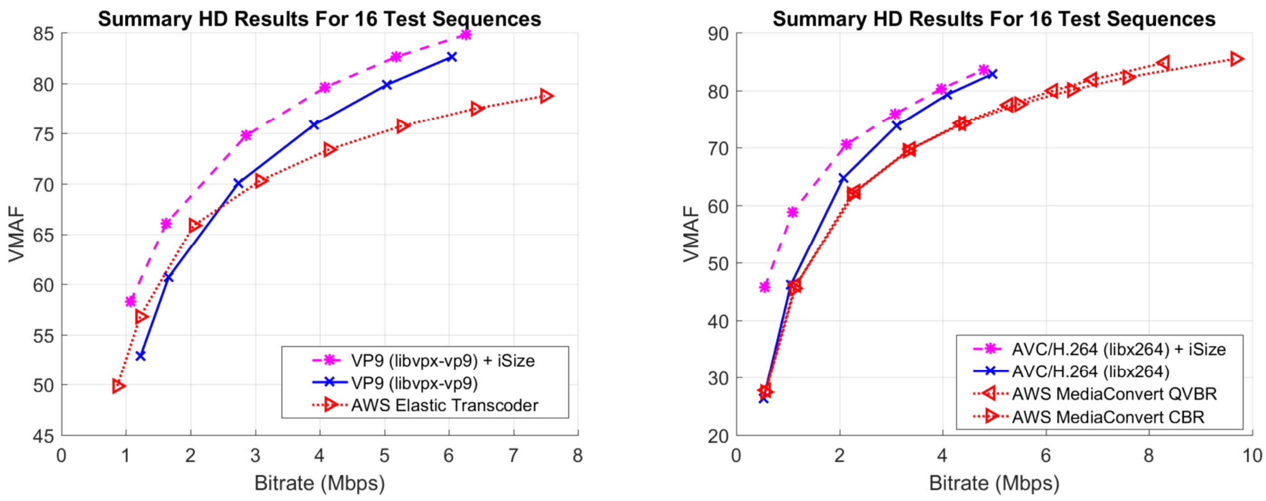

Figure 5. **Left:** VP9 encoding with precoding enabled vs. VP9 standalone and the AWS Elastic Transcoder VP9 codec. **Right:** AVC/H.264 encoding with precoding enabled vs. AVC/H.264 standalone and the AWS MediaConvert H.264/AVC codec.

CONCLUSION

We propose the concept of adaptive precoding for video based on deep 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 at the decoder and display side. 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 realistic HD and UHD video streaming bitrates and state-of-the-art encoder configurations, our precoding offers 14%-55% reduction in bitrate over linear downscaling solutions for AVC/H.264 and HEVC, with substantial potential for further improvement. In addition, for realistic AVC/H.264 and VP9 encoding conditions,

[7] Setup: "slower" preset, GOP=90; since AWS MediaConvert supports VBV encoding via the QVBR mode, we opt for VBV encoding for libx264: https://trac.ffmpeg.org/wiki/Encode/H.264#AdditionalInformationTips with crf=18 for all $s \geq 2$ and crf=23 for all $s < 2$.

precoding is found to offer 23%-43% rate saving over the equivalent encoder. Its compaction features ensure that not only bitrate is saved, but also that video encoding complexity reduction can be achieved. Future work may consider how to extend the notion of precoding beyond DASH/HLS systems, by learning to adaptively preprocess video inputs such that they are optimally recovered by current decoders under specified bitrate ranges of interest.

REFERENCES

[1] J. Kim, *et al.*, "Accurate image superresolution using very deep convolutional networks," *Proc. IEEE Conf. Comput. Vision Pattern Rec.*, pp. 1646–1654, 2016.

[2] A. Wiesel, *et al.*, "Linear precoding via conic optimization for fixed MIMO receivers," *IEEE Trans. Signal Process.*, vol. 54, no. 1, pp. 161-176, 2006.

[3] I. Katsavounidis, "Dynamic optimizer – A perceptual video encoding optimization framework," *The Netflix Tech Blog*, 2018.

[4] C. G. Bampis, *et al.*, "Towards perceptually optimized end-to-end adaptive video streaming," arXiv preprint, *arXiv:1808.03898*, 2018.

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

[6] G. Bjontegaard, "Calculation of av. PSNR diff. betw. RD-curves," *VCEG-M33*, 2001.

[7] B. Lim, *et al.*, "Enhanced deep residual networks for single image super-resolution," *Proc. IEEE Conf. Comput. Vision Pattern Rec.* Worksh., pp. 136–144, 2017.

[8] O. Rippel, *et al.*, "Learned Video Compression," arXiv preprint, *arXiv:1811.06981* (2018).

[9] C. Dong, *et al.*, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. and Machine Intel.*, vol. 38, no. 2, pp. 295–307, Feb. 2016.

[10] S. Ma, *et al.* "Rate-distortion analysis for H. 264/AVC video coding and its application to rate control," *IEEE Trans. CSVT, vol.* 15, no.12, pp. 1533-1544, Dec. 2005.

[11] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *Proc. 13th Int. Conf. Artif. Intel. and Stat.*, pp. 249–256, 2010.

[12] E. Agustsson and R. Timofte, "NTIRE 2017 challenge on single image super-resolution: Dataset and study," *Proc. IEEE CVPR Worksh.*, pp. 126–135, 2017.