Efficient Video Compression via Content-Adaptive Super-Resolution

Mehrdad Khani, Vibhaalakshmi Sivaraman, Mohammad Alizadeh
MIT CSAIL
{khani,vibhaa,alizadeh}@csail.mit.edu

Abstract

Video compression is a critical component of Internet video delivery. Recent work has shown that deep learning techniques can rival or outperform human-designed algorithms, but these methods are significantly less compute and power-efficient than existing codecs. This paper presents a new approach that augments existing codecs with a small, content-adaptive super-resolution model that significantly boosts video quality. Our method, SRVC, encodes video into two bitstreams: (i) a content stream, produced by compressing downsampled low-resolution video with the existing codec, (ii) a model stream, which encodes periodic updates to a lightweight super-resolution neural network customized for short segments of the video. SRVC decodes the video by passing the decompressed low-resolution video frames through the (time-varying) super-resolution model to reconstruct high-resolution video frames. Our results show that to achieve the same PSNR, SRVC requires 20% of the bits-per-pixel of H.265 in slow mode, and 3% of the bits-per-pixel of DVC, a recent deep learning-based video compression scheme. SRVC runs at 90 frames per second on an NVIDIA V100 GPU.

1. Introduction

Recent years have seen a sharp increase in video traffic. It is predicted that by 2022, video will account for more than 80% of all Internet traffic [6, 1]. Video delivery is so bandwidth-intensive that during surge periods such as the initial months of the pandemic, Netflix and Youtube were forced to throttle video quality to reduce overheads [2, 3]. Further, while mobile devices support 1080p resolutions these days, cellular networks are still plagued by low bandwidth and frequent fluctuations in most parts of the world. Hence efficient video compression to reduce bandwidth consumption without compromising on quality is more critical than ever.

While the demand for video content has increased over the years, the techniques used to compress and transmit video have largely remained the same. Ideas such as applying Discrete Cosine Transforms (DCTs) to video blocks and computing motion vectors [46, 18] , which were developed decades ago, are still in use today. Even the latest H.265 codec improves upon these same ideas by incorporating variable block sizes [7]. Recent efforts [35, 10, 38] to improve video compression have turned to deep learning to capture the complex relationships between the components of a video compression pipeline. These approaches have had moderate success at outperforming current codecs, but they are much less compute- and power-efficient.

We present SRVC, a new approach particularly useful for cellular networks and low bitrate-scenarios, that combines existing compression algorithms with a lightweight, content-adaptive super-resolution (SR) neural network that significantly boosts performance with low computation cost. SRVC compresses the input video into two bitstreams: a *content stream* and a *model stream*, each with a separate bitrate that can be controlled independently of the other stream. The content stream relies on a standard codec such as H.265 to transmit low-resolution frames at a low bitrate. The model stream encodes a *time-varying* SR neural network, which the decoder uses to boost the quality of decompressed frames derived from the content stream. SRVC uses the model stream to specialize the SR network for short segments of video dynamically (e.g., every few seconds). This makes it possible to use a small SR model, consisting of just a few convolutional and upsampling layers.

Applying SR to improve the quality of low-bitrate compressed video isn't new. AV1 [16], for instance, has a mode (typically used in low-bitrate settings) that encodes frames at low resolution and applies an upsampler at the decoder. While AV1 relies on standard bicubic [26] or bilinear [52] interpolation for upsampling, recent proposals have shown that learned SR models can significantly improve the quality of these techniques [33, 20].

However, these approaches rely on *generic* SR neural networks [45, 53, 25]) that are designed to generalize across a wide range of input images. These models are large (e.g., 10s of millions of parameters) and can typically reconstruct only a few frames per second even on high-end GPUs [31]. But in many usecases, generalization isn't necessary. In par-

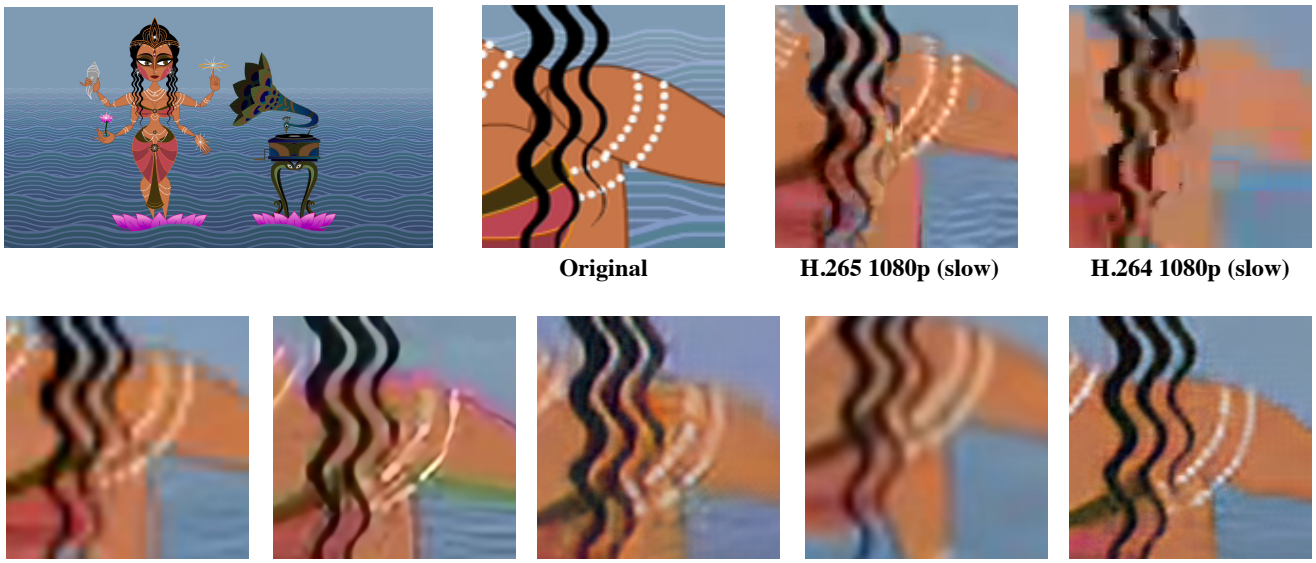

Figure 1: Comparing different video compression schemes at a 200 Kbps bitrate (except for DVC) on the 1560th frame of Sita Sings the Blues video in Xiph [9] dataset. DVC [34] is encoding at its lowest available bitrate that requires 4.97 Mbps in this example.

ticular, we often have access to the video being compressed ahead of time (e.g, for on-demand video). Our goal is to dramatically reduce the complexity of the SR model in such applications by specializing it (in a sense, overfitting it) to short segments of video.

To make this idea work, we must ensure that the overhead of the model stream is low. Even with our small SR model (with 2.22M parameters), updating the entire model every few seconds would consume a high bitrate, undoing any compression benefit from lowering the resolution of the content stream. SRVC tackles this challenge by carefully selecting a small fraction (e.g., 1%) of parameters to update for each segment of the video, using a "gradient-guided" coordinate-descent [48] strategy that identifies parameters that have the most impact on model quality. Our primary finding is that a SR neural network adapted in this manner over the course of a video can provide such a boost to quality, that including a model stream along with the compressed video is more efficient than allocating the entire bitstream to content.

In summary, we make the following contributions:

- We propose a novel dual-stream approach to video streaming that combines a time-varying SR model with compressed low-resolution video produced by a standard codec. We develop a coordinate descent method to update only a fraction of model parameters for each few-second segment of video with low overhead.
- We propose a lightweight model with spatially-adaptive kernels, designed specifically for content-specific SR. Our model runs in real-time, taking only 11 ms (90 fps) to generate a 1080p frame on an NVIDIA V100 GPU. In comparison, DVC [35] takes 100s of milliseconds at the same resolution.
- We show that, in low bitrate regimes, to achieve the same PSNR, SRVC requires only 20% of the bitrate as H.265 in its *slow* encoding mode [1], and 3% of DVC's bits-per-pixel. SRVC's quality improvement extends across all frames in the video.

Figure 1 shows visual examples comparing the SRVC with these baseline approaches at competitive or higher bitrates. Our datasets and code are available at https://github.com/AdaptiveVC/SRVC.git

2. Related Work

Standard codecs. Prior work has widely studied video encoder/decoders (codecs) such as H.264/H.265 [40, 42], VP8/VP9 [12, 37], and AV1 [16]. These codecs rely on hand-designed algorithms that exploit the temporal and spatial redundancies in video pixels, but cannot adapt to specific videos. Existing codecs are particularly effective when used in *slow* mode for offline compression. Nevertheless, SRVC's combination of a low-resolution H.265 stream with a content-adaptive SR model outperforms H.265 at high resolution, even in its *slow* mode. Some codecs like AV1 provide the option to encode at low resolution and upsample using bicubic interpolation [26]. But, as we show in §4, SRVC's learned model provides a much larger improvement

[1] To the authors' knowledge, this is the first learning-based scheme that compares to H.265 in its slow mode

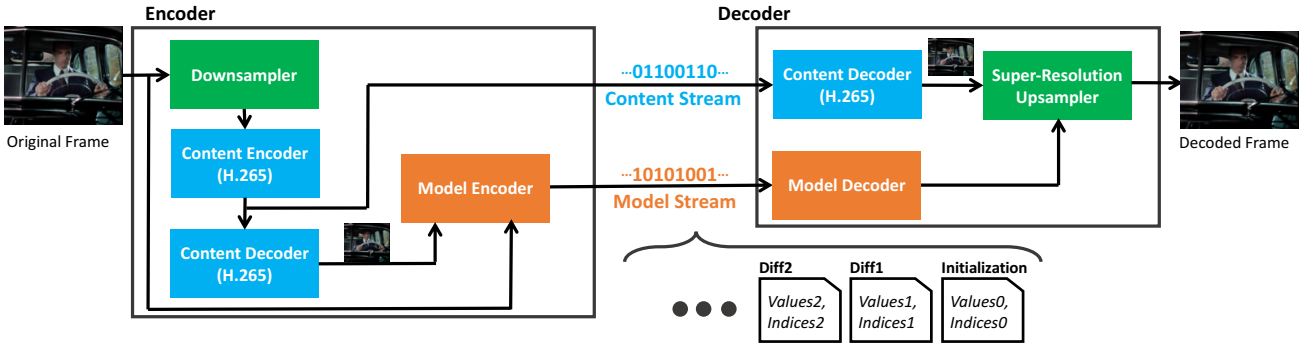

Figure 2: SRVC encodes video into two bitstreams.Content stream encodes downsampled low-resolution video with the existing codec. Model stream encodes periodic updates to a lightweight super-resolution neural network customized for short segments of the video.

in video quality compared to bicubic interpolation.

Super resolution. Recent work on single-image SR [53, 25] and video SR [33, 20, 22, 30] has produced a variety of CNN-based methods that outperform classic interpolation methods such as bilinear [52] and bicubic [26]. Accelerating these SR models has been of interest particularly due to their high computational complexity at higher resolutions [54]. Our design adopts the idea of subpixel convolution [41], keeping the spatial dimension of all layers identical to the low-resolution input until the final layer. Fusing the information from several video frames has been shown to further improve single-image SR models [44]. However, to isolate the effects of using a content-adaptive SR model, we focus on single-image SR in this work.

Learned video compression. End-to-end video compression techniques [38, 34, 10, 51, 50] follow a compression pipeline similar to standard codecs but replace some of the core components with DNN-based alternatives, e.g., flow estimators [19] for motion compensation and autoencoders [21] for residue compression. However, running these models in real time is challenging. For example, even though the model in [38] is explicitly designed for low-latency video compression, it decodes only 10 frames-per-second (fps) for 640×480 resolution on an NVIDIA Tesla V100 [38]. In contrast, H.264 and H.265 process a few hundred frames a second at the same resolution. Moreover, existing learned video compression schemes are designed to generalize and not targeted to specific videos. Few approaches have proposed overfitting [23] and updating only specific layers [29] of the SR model, yet do not go as far as presenting a holistic solution and an extensive evaluation. In this work, we show that augmenting existing codecs with content-adaptive SR achieves better quality and compression than end-to-end learned compression schemes.

Lightweight models. Lightweight models intended for phones and compute-constrained devices have been designed manually [39] and using neural architecture search techniques [55, 49]. Model quantization and weight pruning [24, 32, 14, 17] have helped reduce the computation footprint of models with a small loss in accuracy. Despite the promise of these optimizations, the accuracy of these lightweight models falls short of state-of-the-art solutions. SRVC is complementary to such optimization techniques and would benefit from them.

3. Methods

Figure 2 shows an overview of SRVC's compression pipeline. SRVC compresses video into two bitstreams:

1. **Content stream:** The encoder downsamples the input video frames by a factor of k in each dimension (e.g., k=4) to generate low-resolution (LR) frames using area-based downsampling. It then encodes the LR frames using an off-the-shelf video codec to generate the content bitstream (our implementation uses H.265 [7]). The decoder decompresses the content stream using the same codec to reconstruct the LR frames. Since video codecs are not lossless, the LR frames at the decoder will not exactly match the LR frames at the encoder.

2. **Model stream:** A second bitstream encodes the SR model that the decoder uses to upsample the each decoded LR frame. We partition the input video into N fixed-length segments, each τ seconds long (e.g., $\tau = 5$). For each segment $t \in \{0, ..., N-1\}$, we adapt the SR model to the frames in that segment during encoding. Specifically, the encoder trains the SR model to map the LR decompressed frames within a segment to high-resolution frames. Let $\mathbf{\Theta}_t$ denote the SR model parameters obtained for segment t. The model adaptation is sequential: the training procedure for segment t initializes the model parameters to $\mathbf{\Theta}_{t-1}$. The model stream encodes the sequence $\mathbf{\Theta}_t$ for $t \in \{0, ..., N-1\}$. It starts with the full model $\mathbf{\Theta}_0$, and then encodes the *changes* in the parameters for each subsequent model update, i.e., $\mathbf{\Delta}_t = \mathbf{\Theta}_t - \mathbf{\Theta}_{t-1}$. The decoder updates the parameters every τ seconds, using the last model parameters $\mathbf{\Theta}_{t-1}$ to find $\mathbf{\Theta}_t = \mathbf{\Theta}_{t-1} + \mathbf{\Delta}_t$.

The model stream adds overhead to the compressed bitstream. To reduce this overhead, we develop a small model that is well-suited to *content-specific* SR (§3.1), and design an algorithm that significantly reduces the overhead of model adaptation by training only a small fraction of the model parameters that have the highest impact on the SR quality in each segment (§3.2).

3.1. Lightweight SR Model Architecture

Existing SR models typically use large and deep neural networks (e.g., typical EDSR has 43M parameters across more than 64 layers [31]), making them difficult to use in a real-time video decoder. Moreover, adapting a large DNN model to specific video content and transmitting it to the decoder would incur high overhead.

We propose a new lightweight architecture that keeps the model small and shallow, and yet, is very effective at content-based adaptation (§4.2). Our model is inspired by classical algorithms like bicubic upsampling [27], which typically use only one convolutional layer and a fixed kernel for upsampling the entire image. It uses this basic architecture but replaces the fixed kernel with spatially-adaptive kernels that are customized for different regions of the input frame. Our model partitions each frame into patches, and uses a shallow CNN operating on the patches to generate different (spatially-adaptive) kernels for each patch (Fig. 3).

More formally, the model first partitions an input frame into equal-sized patches of $P \times P$ pixels (e.g. $P = 5$ pixels) using a common space-to-batch operation. For each patch, a patch-specific block (Adaptive Conv Block in Fig. 3) computes a 3×3 convolution kernel with 3 input and F output channels ($27F$ parameters) using a two-layer CNN, and applies this kernel (pink box) to the patch. The forward pass of the adaptive conv block with input patch $\mathbf{x} \in \mathbb{R}^{P \times P \times 3}$ and output features $\mathbf{y} \in \mathbb{R}^{P \times P \times F}$ is summarized as follows:

$$\mathbf{w} = f(\mathbf{x}),$$
$$\mathbf{y} = \sigma(\mathbf{w} * \mathbf{x}).$$

We use a two-layer CNN to model $f(\cdot)$ in our architecture. We finally reassemble the feature patches (batch-to-space) and compute the output using another two-layer CNN followed by a pixel shuffler (depth-to-space) [41] that brings the content to the higher resolution. All convolutions have a kernel height and width of 3, except for the first layer of the regular block that uses kernel size of 5.

3.2. Model Adaptation and Encoding

Training algorithm. We use the L2-loss between the SR model's output and the corresponding high-resolution frame (input to the encoder), over all the frames in each segment to train the model for that segment. Formally, we define the loss as

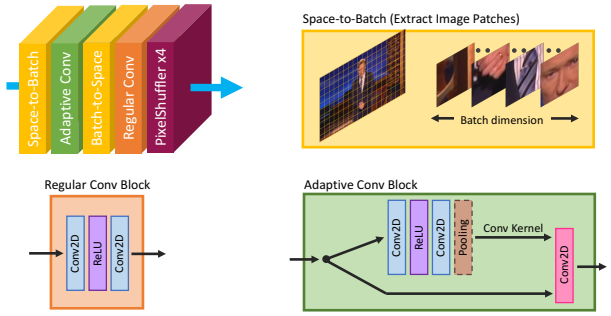

Figure 3: SRVC lightweight SR model architecture.

$$L(\Theta_t) = \frac{1}{n|F_t|} \sum_{i=1}^{n} \sum_{j=1}^{|F_t|} ||Y_{ij} - X_{ij}||^2$$

where $|F_t|$ is the number of frames in the t^{th} segment, each with n pixels, and Y_{ij} and X_{ij} denote the value of the i^{th} pixel in the j^{th} frame of the decoded high-resolution output frame and the original high-resolution input frame respectively. During the training, we randomly crop the samples at half of their size in each dimension. We use Adam optimizer [28] with learning rate of 0.0001, and first and second momentum decay rates of 0.9 and 0.999.

To reduce the model stream bitrate, we update only a fraction of the model parameters across video segments. Our approach is to update only those parameters that have the most impact on the model's accuracy. Specifically, we update the model parameters with *the largest gradient magnitude* for each new segment as follows. First, we save a copy of the model at the beginning of a new segment and perform one iteration of training over all the frames in the new segment. We then choose the fraction η of the parameters with the largest magnitude of change in this iteration, and reset the model parameters to the starting saved copy. We apply the Adam updates for only the selected paramaters and discard the updates for the rest of the model (keeping those parameters fixed).

Encoding the model stream. To further compress the model stream, we only transmit changes to the model parameters at each update. We encode the model updates into a bitstream by recording the indices and associated change in values of the model parameters (Fig. 2). SRVC's model encoding is lossless: the encoder and decoder both update the same subset of parameters during each update. To update a fraction η of the parameters for a model with M float16 parameters, we need an average bitrate of at most $(16+\log(M)) \times \eta M/\tau$ to express the deltas and the indices every τ seconds. For example, with model size $M = 2.22$ million parameters (F=32, see Table 2), $\tau = 10$ seconds, and $\eta = 0.01$, we only require 82 Kbits/sec to encode the model stream required to generate 1080p video. To put this

number into perspective, Netflix recommends a bandwidth of 5 Mbits/sec at 1080p resolution [4]. The model stream can be compressed further using lossy compression techniques or by dynamically varying η or the model update frequency based on scene changes.

Training the SR model for 1080p resolution and encoding the updates into the model stream takes about 12 minutes for each minute worth of video with our un-optimized implementation. However, given the small compute overhead of our lightweight model, we shared a V100 GPU between five simultaneous model training (encoding) processes without any significant slow down to any process. Hence, the overall throughput of the encoding on V100 GPU is about 2.5 minutes of training per minute of content. We consider this duration feasible for offline compression scenarios where videos are available to content providers well ahead of viewing time. We believe that there is significant room to accelerate the encoding process too with standard techniques (e.g., training on sampled frames rather than all frames) and further engineering. We leave an exploration of these opportunities to future work.

4. Experiments

4.1. Setup

Dataset. Video datasets like JCT-VC [15], UVG [36] and MCL-JCV [43], consisting of only a few hundred frames ($\sim$10 sec) per video, are too short to evaluate SRVC's content-adaptive SR. Hence, we train and test the efficacy of SRVC on a custom dataset consisting of 28 downloadable videos from Vimeo (short films) and 4 full-sequence videos from the Xiph Dataset [9]. We trim all videos to 10 minutes and resize them to 1080p resolution in RAW format from their original 4K resolution and MPEG-4 format using area-based interpolation [47]. We use the resulting 1080p frames as our high-resolution source frames in our pipeline. We re-encode each video's raw frames at different qualities or Constant Rate Factors (CRFs) on H.264/H.265 to control the bitrate. We also use area-interpolation to downsample the video to 480p and encode the low-resolution (LR) video using H.265 at different CRFs to achieve different degrees of compression. The SR model in SRVC is then trained to learn the mapping from each LR video at a particular CRF to the original 1080p video at its best quality.

Baselines. We compare the following approaches. The first four only use a content stream while the next three use both a content stream and a model stream. The last approach is an end-to-end neural compression scheme.

- **1080p H.264/H.265:** We use ffmpeg and the libx264/libx265 codec to re-encode each of the 1080p videos at different CRFs using the *slow* preset.
- **480p H.265 + Bicubic upsampling:** We use ffmpeg and the libx265 codec to downsample the 1080p original video to LR 480p at different CRFs using area-interpolation and the *slow* preset. This approach's bitrate comes only from its content stream: the downsampled 480p frames encoded using H.265. We use bicubic interpolation to upsample the 480p videos back to 1080p. This isolates the bitrate reduction from just encoding at lower resolutions.
- **480p H.265 + Generic SR:** Instead of Bicubic upsampling, we use a more sophisticated DNN-based super-resolution model (EDSR [31] with 16 residual blocks) to upsample the 480p frames to 1080p. The upsampling takes about 50ms for each frame (about 5$\times$ worse than SRVC). We use a pre-trained checkpoint that has been trained on a generic corpus of images [11]. Since we expect all devices to be able to pre-fetch such a model, this approach only has a content stream at 480p encoded using H.265. Thus, its bits-per-pixel value is identical to the Bicubic case.
- **480p H.265 + One-shot Customization:** We evaluate a version of SRVC that uses a lightweight SR model (§3.1) without the model adaptation procedure. For this, we train our SR model exactly once (one-shot) using the entire 1080p video and encode it in the model stream right at the beginning before any LR content. The content stream for this approach comprises of the 480p H.265 video while the model stream consists of a single initial model customized to the entire video duration. The overhead of the model is amortized over the entire video and added to the content bitrate when computing the total bits-per-pixel value.
- **480p H.265 + SRVC :** We evaluate SRVC which uses the same initial SR model as One-shot Customization but is periodically adapted to the most recent 5 second segment of the video. To train this model, we use random crops (half the frame size in each dimension) from each reference frame within a video segment. The content stream for SRVC relies on standard H.265. The model stream, on the other hand, is updated every 5 seconds and is computed using our gradient-guided strategy, which only encodes the change to those parameters that have the largest gradients in each video segment (§3.2). To compute the total bits-per-pixel, we add the model stream's bitrate (computed as described in §3.2) to the content stream's bitrate. We also add the overhead of sending the initial model in full to the model stream's bitrate.
- **DVC:** An official checkpoint [5] of Deep Video Compression [35], an end-to-end neural network based compression algorithm. To evaluate DVC, we compute the PSNR and SSIM metrics, and use Lu *et al.*'s[35] estimator to measure their required bits-per-pixel for every frame at four different bitrate-distortion trade-off operating points ($\lambda \in \{256, 512, 1024, 2048\}$).

Model and training procedure. Our model uses 32 output

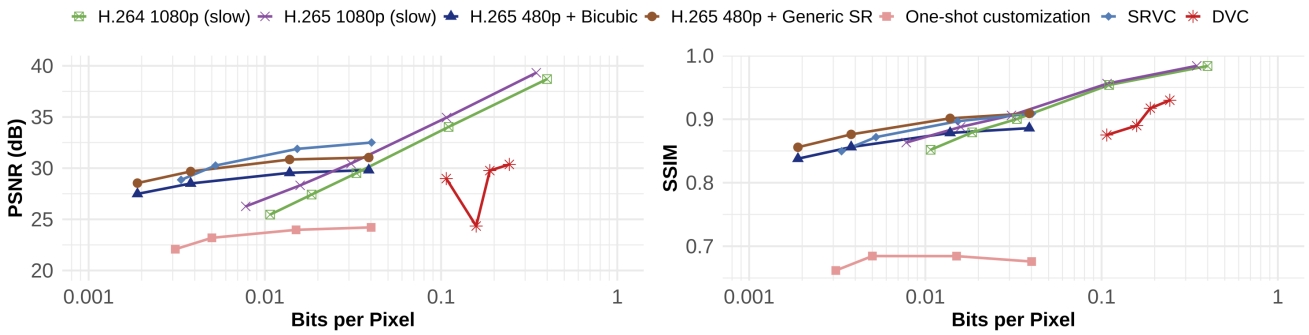

Figure 4: Tradeoff between video quality and bits-per-pixel for different approaches on three long videos from the Xiph dataset. SRVC with content-adaptive streaming reduces the bitrate consumption to 16% of current codecs and ∼2% of end-to-end compression schemes like DVC. Though comparable in video quality to SRVC, the generic SR approach does not run in real-time.

feature channels in the adaptive convolution block, resulting in 2.22 million parameters. However, only 1% of them are updated by the model stream and that too, only every 5 seconds. We vary the number of output feature channels, the fraction of model parameters updated, and the update interval to understand its impact on SRVC's performance.

Metrics and color space. We compute the average Peak Signal-To-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) across all frames at the output of the decoder (after upsampling). We report PSNR based on the mean square error across all pixels in the video (over all frames) where the pixel-wise error itself is computed on the RGB space. SSIM is computed as the average SSIM between the decoded frames and their corresponding high-resolution original counterparts. However, since variations in frame quality over the course of a video can have significant impact on users' experience, we also show a CDF of both PSNR and SSIM across all frames in the video.

We compute the content bitrate for the all approaches relying on H.264/5 at both 1080p and 480p using ffmpeg. For approaches that stream a model in addition to video frames, we compute the model stream bitrate based on the total number of model parameters, the fraction of them that are streamed in each update interval, and the frequency of updates (§3.2). The content and model stream bitrates are combined to compute a single bits-per-pixel metric. Note that the bits-per-pixel range in our evaluations is an order-of-magnitude lower than results reported in prior work [35, 10] because our approach is designed for low-bitrate scenarios and we compare to the *slow mode* in H.264/5 which is more efficient than the "fast" and "medium" modes We plot PSNR and SSIM metrics at different bits-per-pixel to compare different schemes. Since SRVC runs inference on decoded frames as they are rendered to users, its SR model needs to run in real-time. To evaluate its feasibility, we also compare SRVC's speed in frames per second to other learning-based approaches.

Method	BD-PSNR (dB)	BD-Rate (%)
DVC	-10.04	598.76
H.264 1080p (slow)	-1.38	45.5
H.265 1080p (slow)	0	0
H.265 480p (slow) + Bicubic	+0.67	-55.81
H.265 480p (slow) + Generic SR	+2.61	-75.31
SRVC (Ours)	**+3.41**	**-80.09**

Table 1: BD-PSNR and BD-Rate of different approaches on Xiph dataset relative to H.265 1080p (slow).

4.2. Results

Compression performance. Fig. 1 shows a visual comparison of the different schemes for similar bits-per-pixel values. For DVC in this figure, we show the results for the lowest bitrate model available that ends up using 4.97 Mbps, which is significantly larger than the 200 Kbps bitrate of other schemes in this example. To compare the compression provided by different approaches across a wide range of bits-per-pixel values, we analyze the PSNR and SSIM achieved by different methods on three long Xiph [9] videos in Fig. 4. Tab. 1 summarizes the BD-Rate and BD-PSNR [13] metrics for the same experiment. Note that the bits-per-pixel metric captures both the contribution of the content and the model for those approaches that use a model stream for SR. We do not report the bitrate distortion metrics for One-shot customization as its PSNR hardly overlaps with H.265.

As seen in Fig. 4, SRVC achieves PSNR comparable to today's H.265 standard (in *slow* mode) with far less bits-per-pixel. For instance, to achieve a PSNR of 30 dB, SRVC requires only 0.005 bits-per-pixel while H.265 and H.264 codecs, even in their slowest settings, require more than 0.03 bits-per-pixel. In BD-Rate and BD-PSNR terms (Tab. 1), SRVC on average achieves a 3.41dB improvement relative to H.265 slow preset at 1080p at the same bitrate, or requires only 20% of the bitrate to achieve the same PSNR. However, One-shot Customization's performs poorer than a simple bicubic interpolation. This is because

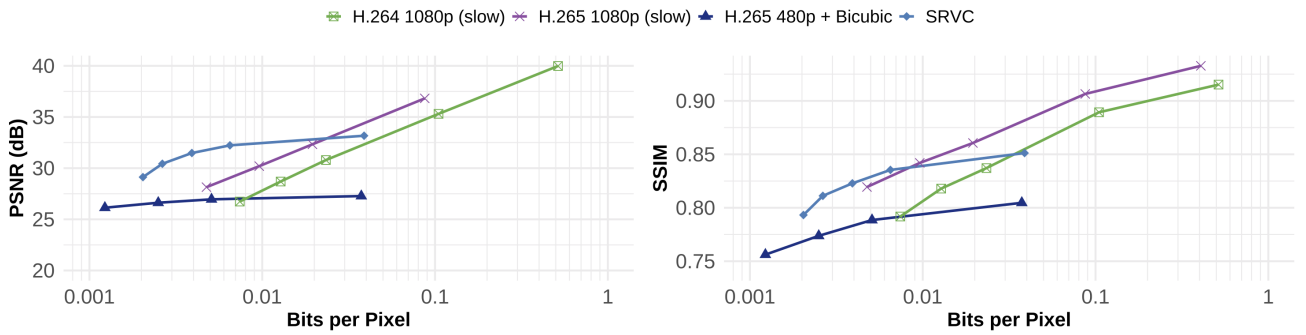

Figure 5: Tradeoff between video quality and bits-per-pixel for different approaches on 28 videos from Vimeo. To achieve 30dB PSNR, SRVC requires 10% and 25% of the bits-per-pixel required by H.264 and H.265 in their *slow* modes.

SRVC's custom SR model is not large enough to generalize to the entire video, but has enough parameters to learn a small segment. It is worth noting that to achieve the same PSNR, SRVC requires only 3% of the bits-per-pixel required by DVC [35], the end-to-end neural compression scheme. SRVC 's SSIM is comparable but 0.01-0.02 better than current codecs for the same level of bits-per-pixel, particularly at higher bitrates. SRVC also outperforms a generic SR approach (EDSR) by 0.8dB and 4.8% respectively on BD-Rate and BD-PSNR metrics.

Fig. 4 suggests that a 480p stream augmented with a generic SR model performs just as well as SRVC in terms of its PSNR and SSIM for a given bits-per-pixel level. However, typical SR models are too slow to perform inference on a single frame (about 5× slower in this case), making them unfit for real-time video delivery. To evaluate the performance of viable schemes on real-world video, we evaluate the bits-per-pixel vs. video quality tradeoff on 28 videos publicly available on Vimeo. As Fig. 5 suggests, SRVC outperforms all other approaches on the PSNR achieved for a given bits-per-pixel value. In particular, to achieve 30dB PSNR, SRVC requires 25% and 10% of the bits-per-pixel required by H.265 and H.264 respectively.

A key takeaway from Tab. 1, and Figures. 4 and 5 is that for a given bitrate budget, SRVC achieves better quality than standard codecs. This suggests that beyond a baseline bitrate for the content, it is better to allocate bits to streaming a SR model than to dedicate more bits to the content. We describe this trade-off between model and content bitrates in more detail in Fig. 7.

Robustness of quality improvements. To see if SRVC's improvements come from just producing a few high-quality frames right after the model is updated, we plot a CDF of the PSNR and SSIM values across all frames of the Meridian video in Fig. 6. We compare schemes at a bits-per-pixel value of ∼0.002. Since DVC [35] has a much higher bits-per-pixel and EDSR [31] performs poorly on this video, we exclude both approaches [2]. Firstly, we notice that both One-shot Customization and SRVC perform better than other schemes. Further, this improvement occurs over all of the frames in that no frame is worse off with SRVC than it is with the defacto codec. In fact, over 50% of the frames experience a 2–3 dB improvement in PSNR and a 0.05–0.0075 improvement in SSIM with both versions of SRVC.

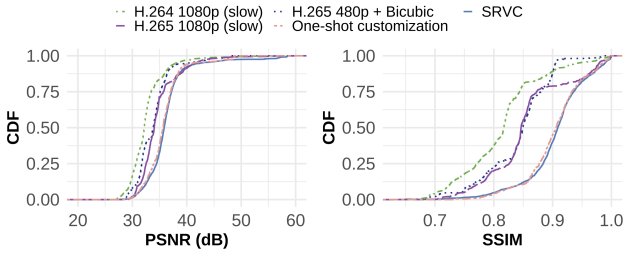

Figure 6: CDF of PSNR and SSIM improvements with SRVC across all video frames at a bits-per-pixel of 0.002. The quality enhancement from SRVC is not limited to only those frames that follow a model update.

#Feature Channels (F)	8	16	32	64	128
PSNR(dB)	38.49	38.69	39.87	39.89	39.90
SSIM	0.942	0.944	0.946	0.947	0.949
Inference Time (ms)	7	9	11	17	25
Num. of Parameters	0.59M	1.14M	2.22M	4.39M	8.72M

Table 2: Impact of number of output feature channels in SRVC's adaptive convolutional block on inference time and quality metrics for a video snippet on an NVIDIA V100 GPU.

Impact of number of Output Feature Channels. Since SRVC downsamples frames at the encoder and then streams a model to the receiving client who resolves the decoded frames, it is important that SRVC performs inference fast enough to run at the framerate of the video on an edge-device with limited processing power. Viewers need at least 30 fps for good quality. Consequently, the inference time on a single frame cannot afford to be longer than 33ms. In fact, the Meridian [8] video has a frame rate of 60 fps, so

[2]Figures. 4, 5 and 6 cover different videos, and thus, their results cannot be directly compared.

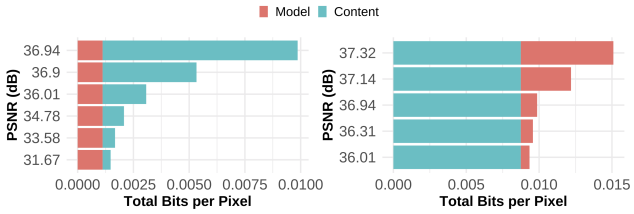

Figure 7: Impact of varying bits-per-pixel for the content stream for a fixed model bitrate and vice-versa. Increasing the bits-per-pixel for the low-resolution H.265 content stream improves PSNR, especially at low bitrates. At higher content bitrates, increasing the model bitrate by transmitting more model parameters further improves PSNR.

running low-latency inference is even more critical.

To evaluate the practicality of SRVC's lightweight model, we evaluate the end-to-end inference time per frame on an NVIDIA V100 GPU as we vary the number of the output feature channels in the adaptive convolution block (F) in Tab. 2. While increasing F improves the PSNR and SSIM values due to better reconstruction of the fine details, it comes at a cost. With $F = 64$ and $F = 128$, the inference times of 17 ms and 25 ms respectively causing the frame rate to drop below the input 60 fps. Further, the number of parameters increases to nearly $10M$, a steep number for the model to stream periodically. Hence, we design SRVC's model to use 32 output feature channels, ensuring it takes only 11 ms to run inference on a single frame. In comparison, the EDSR generic SR model is about $5\times$ slower to perform inference on a single frame. Even the end-to-end neural video compression approach DVC [35] takes over hundreds of milliseconds to infer a single frame at 1080p.

Trade-off between model bitrate and content bitrate in SRVC. The presence of dedicated model and content streams in SRVC implies that the bitrate for each stream can be controlled independent of the other, to achieve different compression levels. Fig. 7 shows the impact of altering the content bitrate for a fixed model bitrate and vice-versa, when encoding the Meridian video using SRVC. The content bits-per-pixel is varied by changing the quality (CRF) of the 480p H.265 stream. In contrast, the contribution from the model bits-per-pixel is controlled by the fraction of model parameters transmitted during each update.

As anticipated, for a fixed amount of model bits-per-pixel (updating 1% of the model parameters), PSNR improves as the content bitrate is increased. This is because as the quality of the underlying low-resolution H.265 frames improves, it becomes easier for the model to resolve them to their 1080p counterparts. Increasing the content bitrate from the lowest quality level of CRF 35 (with 0.0014 bits-per-pixel) to CRF 20 (with 0.003 bits-per-pixel) improves PSNR from 31 dB to 36 dB. However, increasing the bits-per-pixel for the content beyond that yields diminishing returns on PSNR (also illustrated in Fig. 5). At higher quality levels, Fig. 7 suggest that modest increases in the bits-per-pixel allocated to the model result in large improvements to the PSNR. For instance, adapting 10% of the model parameters consumes 0.006 bits-per-pixel, 6x more bits-per-pixel than adapting 0.5% of the model parameters, but results in a PSNR improvement of 1dB from 36.31 dB to 37.32 dB.

Update Interval (s)	**5**	**10**	**15**	**20**	∞
PSNR(dB)	37.25	36.52	36.57	36.45	35.32
SSIM	0.92	0.91	0.91	0.91	0.91
Bits-per-pixel	0.006	0.003	0.002	0.0015	0

Table 3: Impact of SRVC's model update interval on the bits-per-pixel consumed by model updates and the associated gains in video quality. We find that an update interval of 5 seconds strikes a good trade-off between bits-per-pixel and quality.

Impact of SRVC's update interval. SRVC can also control the bits-per-pixel consumed by the model stream by varying the interval over which updates to the SR model are performed. Frequent updates increase the model bitrate, but ensure better reconstruction since the model is trained on frames very similar to the current frame. An extreme scenario is an update interval of ∞ that corresponds to the One-shot Customization. Tab. 3 captures the impact of varying the update interval on the average quality of decoded frames from the Meridian video. We find that an update interval of 5 seconds achieves good performance without compromising much on bits-per-pixel. The fact that the PSNR does not degrade significantly for modest increases to the update interval suggests further optimizations atop SRVC that only update the model after a drastic scene change.

5. Conclusion

In this work, we present SRVC, an approach that augments existing video codecs with a lightweight and content-adaptive super-resolution model. SRVC achieves video quality comparable to modern codecs with better compression. Our design is a first step towards leveraging super-resolution as a video compression technique. Future work includes further optimizations to identify the pareto frontier for the model vs. content bitrate trade-off, more sophisticated techniques to detect scene changes and optimize update intervals, and the design of more effective lightweight super-resolution neural network architectures.

6. Acknowledgments

We would like to thank our anonymous reviewers and meta-reviewer for their valuable feedback. This work was supported in part by NSF grants CNS-1751009, CNS-1955370, CNS-1910676, a Cisco Research Center Award, a Microsoft Faculty Fellowship, and awards from the MachineLearningApplications@CSAIL and MIT.nano NCSOFT programs.

References

[1] https://www.forbes.com/sites/markbeech/2020/03/25/covid-19-pushes-up-internet-use-70-streaming-more-than-12-first-figures-reveal/?sh=4335cc443104. 1

[2] https://abcnews.go.com/Technology/netflix-youtube-throttle-streaming-quality-europe-coronavirus-forces/story?id=69754458. 1

[3] https://www.theverge.com/2020/3/20/21188072/amazon-prime-video-reduce-stream-quality-broadband-netflix-youtube-coronavirus. 1

[4] https://help.netflix.com/en/node/306. 5

[5] https://github.com/GuoLusjtu/DVC/tree/master/TestDemo/VideoCodec/model. 5

[6] "cisco annual internet report (2018–2023) white paper". https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html. 1

[7] x265 HEVC Encoder / H.265 Video Codec. http://x265.org/. 1, 3

[8] Xiph Dataset Meridian Video. https://media.xiph.org/video/derf/meridian/MERIDIAN_SHR_C_EN-XX_US-NR_51_LTRT_UHD_20160909_OV/. 7

[9] Xiph.org Video Test Media. https://media.xiph.org/video/derf/. 2, 5, 6

[10] Eirikur Agustsson, David Minnen, Nick Johnston, Johannes Balle, Sung Jin Hwang, and George Toderici. Scale-space flow for end-to-end optimized video compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8503–8512, 2020. 1, 3, 6

[11] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 126–135, 2017. 5

[12] Jim Bankoski, Paul Wilkins, and Yaowu Xu. Technical overview of vp8, an open source video codec for the web. In *2011 IEEE International Conference on Multimedia and Expo*, pages 1–6. IEEE, 2011. 2

[13] Gisle Bjontegaard. Calculation of average psnr differences between rd-curves. *VCEG-M33*, 2001. 6

[14] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? *arXiv preprint arXiv:2003.03033*, 2020. 3

[15] Frank Bossen et al. Common test conditions and software reference configurations. In *JCTVC-L1100*, volume 12, 2013. 5

[16] Yue Chen, Debargha Murherjee, Jingning Han, Adrian Grange, Yaowu Xu, Zoe Liu, Sarah Parker, Cheng Chen, Hui Su, Urvang Joshi, et al. An overview of core coding tools in the av1 video codec. In *2018 Picture Coding Symposium (PCS)*, pages 41–45. IEEE, 2018. 1, 2

[17] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *ArXiv*, abs/1710.09282, 2017. 3

[18] Guy Cote, Berna Erol, Michael Gallant, and Faouzi Kossentini. H. 263+: Video coding at low bit rates. *IEEE Transactions on circuits and systems for video technology*, 8(7):849–866, 1998. 1

[19] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015. 3

[20] Longtao Feng, Xinfeng Zhang, Xiang Zhang, Shanshe Wang, Ronggang Wang, and Siwei Ma. A dual-network based super-resolution for compressed high definition video. In *Pacific Rim Conference on Multimedia*, pages 600–610. Springer, 2018. 1, 3

[21] Amirhossein Habibian, Ties van Rozendaal, Jakub M Tomczak, and Taco S Cohen. Video compression with rate-distortion autoencoders. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7033–7042, 2019. 3

[22] Muhammad Haris, Gregory Shakhnarovich, and Norimichi Ukita. Recurrent back-projection network for video super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3897–3906, 2019. 3

[23] Gang He, Chang Wu, Lei Li, Jinjia Zhou, Xianglin Wang, Yunfei Zheng, Bing Yu, and Weiying Xie. A video compression framework using an overfitted restoration neural network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 148–149, 2020. 3

[24] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–800, 2018. 3

[25] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017. 1, 3

[26] Robert Keys. Cubic convolution interpolation for digital image processing. *IEEE transactions on acoustics, speech, and signal processing*, 29(6):1153–1160, 1981. 1, 2, 3

[27] Robert Keys. Cubic convolution interpolation for digital image processing. *IEEE transactions on acoustics, speech, and signal processing*, 29(6):1153–1160, 1981. 4

[28] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4

[29] Yat-Hong Lam, Alireza Zare, Francesco Cricri, Jani Lainema, and Miska M Hannuksela. Efficient adaptation of neural network filter for video compression. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 358–366, 2020. 3

[30] Sheng Li, Fengxiang He, Bo Du, Lefei Zhang, Yonghao Xu, and Dacheng Tao. Fast spatio-temporal residual network for

video super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10522–10531, 2019. 3

[31] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 136–144, 2017. 1, 4, 5, 7

[32] Darryl Lin, Sachin Talathi, and Sreekanth Annapureddy. Fixed point quantization of deep convolutional networks. In *International Conference on Machine Learning*, pages 2849–2858, 2016. 3

[33] Hongwei Lin, Xiaohai He, Linbo Qing, Qizhi Teng, and Songfan Yang. Improved low-bitrate hevc video coding using deep learning based super-resolution and adaptive block patching. *IEEE Transactions on Multimedia*, 21(12):3010–3023, 2019. 1, 3

[34] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. Dvc: An end-to-end deep video compression framework. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11006–11015, 2019. 2, 3

[35] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. Dvc: An end-to-end deep video compression framework. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11006–11015, 2019. 1, 2, 5, 6, 7, 8

[36] Alexandre Mercat, Marko Viitanen, and Jarno Vanne. Uvg dataset: 50/120fps 4k sequences for video codec analysis and development. In *Proceedings of the 11th ACM Multimedia Systems Conference*, pages 297–302, 2020. 5

[37] Debargha Mukherjee, Jingning Han, Jim Bankoski, Ronald Bultje, Adrian Grange, John Koleszar, Paul Wilkins, and Yaowu Xu. A technical overview of vp9—the latest open-source video codec. *SMPTE Motion Imaging Journal*, 124(1):44–54, 2015. 2

[38] Oren Rippel, Sanjay Nair, Carissa Lew, Steve Branson, Alexander G Anderson, and Lubomir Bourdev. Learned video compression. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3454–3463, 2019. 1, 3

[39] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. 3

[40] Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. Overview of the scalable video coding extension of the h. 264/avc standard. *IEEE Transactions on circuits and systems for video technology*, 17(9):1103–1120, 2007. 2

[41] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883, 2016. 3, 4

[42] Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on circuits and systems for video technology*, 22(12):1649–1668, 2012. 2

[43] Haiqiang Wang, Weihao Gan, Sudeng Hu, Joe Yuchieh Lin, Lina Jin, Longguang Song, Ping Wang, Ioannis Katsavounidis, Anne Aaron, and C-C Jay Kuo. Mcl-jcv: a jnd-based h. 264/avc video quality assessment dataset. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 1509–1513. IEEE, 2016. 5

[44] Xintao Wang, Kelvin CK Chan, Ke Yu, Chao Dong, and Chen Change Loy. Edvr: Video restoration with enhanced deformable convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019. 3

[45] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018. 1

[46] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h.264/avc video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576, 2003. 1

[47] Ping Wah Wong and Cormac Herley. Area based interpolation for image scaling, Mar. 30 1999. US Patent 5,889,895. 5

[48] Stephen J. Wright. Coordinate descent algorithms. *Mathematical Programming*, 151:3–34, 2015. 2

[49] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10734–10742, 2019. 3

[50] Chao-Yuan Wu, Nayan Singhal, and Philipp Krahenbuhl. Video compression through image interpolation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 416–431, 2018. 3

[51] Ren Yang, Fabian Mentzer, Luc Van Gool, and Radu Timofte. Learning for video compression with hierarchical quality and recurrent enhancement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6628–6637, 2020. 3

[52] Egor Zakharov, Aleksei Ivakhnenko, Aliaksandra Shysheya, and Victor Lempitsky. Fast bi-layer neural synthesis of one-shot realistic head avatars. In *European Conference on Computer Vision*, pages 524–540. Springer, 2020. 1, 3

[53] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 1, 3

[54] Zhang, Zhengdong and Sze, Vivienne. FAST: A Framework to Accelerate Super-Resolution Processing on Compressed Videos. In *CVPR Workshop on New Trends in Image Restoration and Enhancement*, 2017. 3

[55] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016. 3