Supplemental Material: Conditional Entropy Coding for Efficient Video Compression

Jerry Liu[1], Shenlong Wang[1,2], Wei-Chiu Ma[1,3], Meet Shah[1],
Rui Hu[1], Pranaab Dhawan[1], Raquel Urtasun[1,2]

Uber ATG[1], University of Toronto[2], MIT[3]
{jerryl, slwang, weichiu, meet.shah, rui.hu, pdhawan, urtasun}@uber.com

Abstract. In this supplementary material, we first showcase additional qualitative results in the attached video. We then showcase additional quantitative results on two additional evaluation datasets, MCL-JVC and VTL, as well as additional runtime results. Moreover, we provide additional neural network architecture details for the encoder/decoder and the conditional entropy module. We then conduct some additional quantitative ablation analysis: showcasing effects of internal learning steps on rate-distortion performance, effects of GoP sizes in video codec performance, and effects of input frames on performance.

1 Additional Qualitative Results

We first provide additional qualitative results in the form of a video, attached as `eccv2020_supp.mp4`. In the video, we highlight our compression framework vs. H.265 *veryslow* on 3 low-framerate video sequences - the first two 12Hz UVG video and the last one 10Hz NorthAmerica video. Our approach outperforms H.265 in these settings in MS-SSIM while achieving a lower bitrate. Qualitatively, we can see that while H.265 tends to introduce motion artifacts within various frames in these low framerate settings, our approach preserves a more even quality of detail within each frame.

Additionally, we provide a few more frame comparisons of our approach vs. H.265 *veryslow* and H.264 *veryslow* in Fig. 1. We see a similar pattern as demonstrated in the video. Because our reconstructions contain less variance in detail quality, they also contain fewer artifacts compared to H.265 and H.264. We do note, however, that H.265 / H.264 tends to assign more bits to certain high frequency details, such as text, providing slightly sharper outputs than our approach.

2 Evaluations on Additional Datasets

We run additional evaluations on two datasets, MCL-JVC [6] and Video Trace Library (VTL) [2]. MCL-JVC is a video benchmarking dataset consisting of 1920x1080 video frames ranging from 24-30fps. VTL is a video benchmark dataset

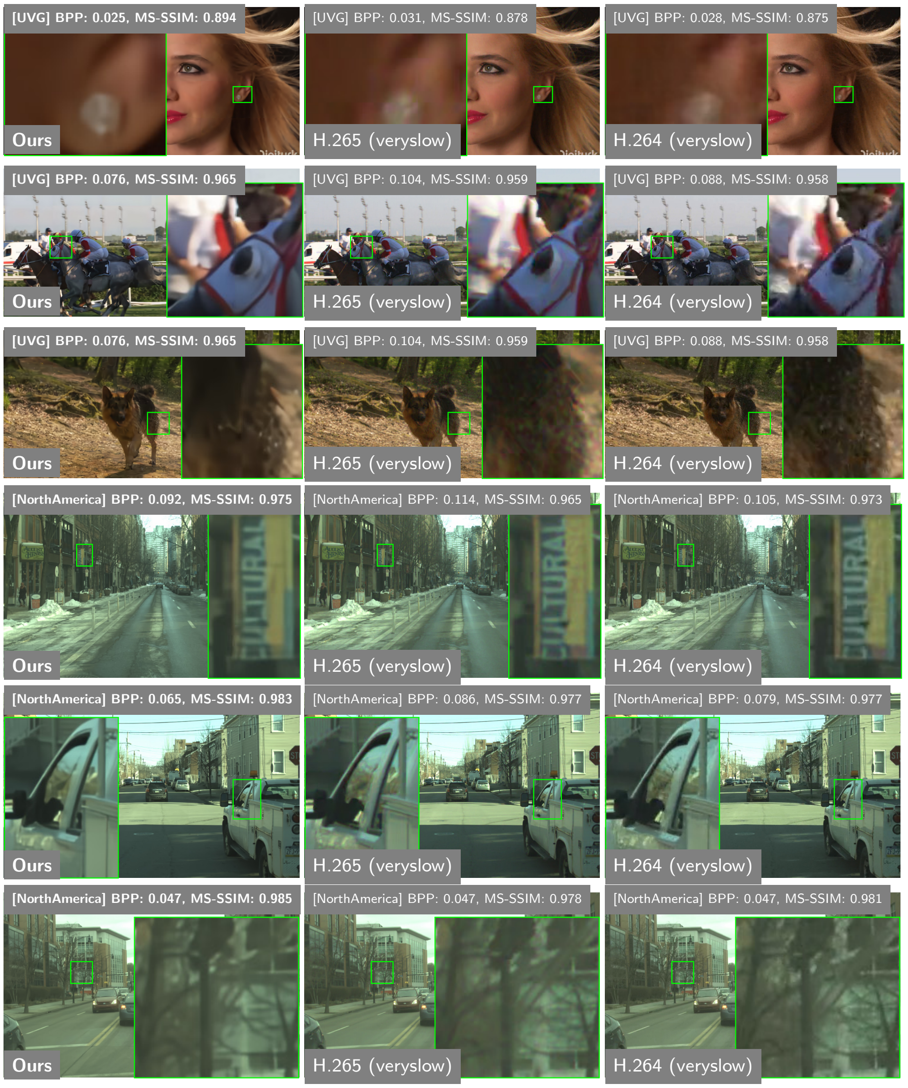
[UVG] BPP: 0.025, MS-SSIM: 0.894
Ours
[UVG] BPP: 0.031, MS-SSIM: 0.878
H.265 (veryslow)
[UVG] BPP: 0.028, MS-SSIM: 0.875
H.264 (veryslow)
[UVG] BPP: 0.076, MS-SSIM: 0.965
Ours
[UVG] BPP: 0.104, MS-SSIM: 0.959
H.265 (veryslow)
[UVG] BPP: 0.088, MS-SSIM: 0.958
H.264 (veryslow)
[UVG] BPP: 0.076, MS-SSIM: 0.965
Ours
[UVG] BPP: 0.104, MS-SSIM: 0.959
H.265 (veryslow)
[UVG] BPP: 0.088, MS-SSIM: 0.958
H.264 (veryslow)
[NorthAmerica] BPP: 0.092, MS-SSIM: 0.975
Ours
[NorthAmerica] BPP: 0.114, MS-SSIM: 0.965
H.265 (veryslow)
[NorthAmerica] BPP: 0.105, MS-SSIM: 0.973
H.264 (veryslow)
[NorthAmerica] BPP: 0.065, MS-SSIM: 0.983
Ours
[NorthAmerica] BPP: 0.086, MS-SSIM: 0.977
H.265 (veryslow)
[NorthAmerica] BPP: 0.079, MS-SSIM: 0.977
H.264 (veryslow)
[NorthAmerica] BPP: 0.047, MS-SSIM: 0.985
Ours
[NorthAmerica] BPP: 0.047, MS-SSIM: 0.978
H.265 (veryslow)
[NorthAmerica] BPP: 0.047, MS-SSIM: 0.981
H.264 (veryslow)

Fig. 1: Additional qualitative demonstration of our approach vs H.265 / H.264 on 12 Hz 1920 × 1080 UVG video and 10 Hz 1920 × 1200 NorthAmerica video.

consisting of lower resolution video frames (352x288). Due to the wide discrepancy of video lengths in the VTL dataset, we set the maximum video length to 300. We run our standard set of video codec baselines on the datasets (H.265 veryslow, H.265 medium, H.264 veryslow), and set the max Group of Pictures (GoP) size to the length of the video.

We plot rate-distortion curves on MCL-JVC, and demonstrate that we outperform video codecs while remaining competitive with the state-of-the-art prior work of Djelouah et al. [5], which utilizes bi-directional interpolation to depend on both the past and future.

We also evaluate our approach on VTL, which yields surprising observations. A cursory look at the rate-distortion plots show that our approach far underperforms those of other video codecs, especially on PSNR. However, a closer analysis of the qualitative results show high-frequency artifacts in the source video that our frame encoder does not capture. We discuss these results and offer an explanation for the quantitative discrepancy of our approach with other codecs below.

2.1 Rate-Distortion Curve on MCL-JVC

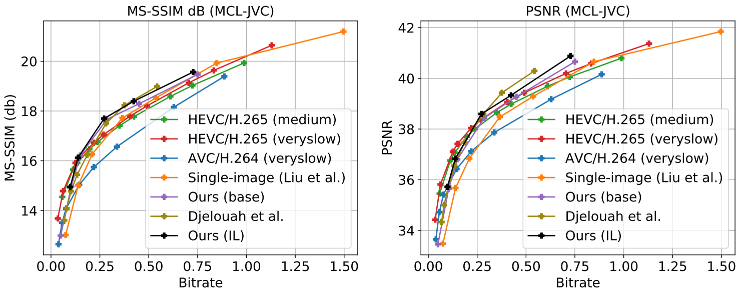

Fig. 2: Rate-distortion plot of our approach vs. competing baselines on MCL-JVC.

We plot the rate-distortion curve on MCL-JVC, as shown in Fig. 2. We observe that our approach outperforms other video codec baselines. Surprisingly, it is also competitive with the state-of-the-art work Djelouah et al. [5]. This is interesting because Djelouah et al. utilizes a bidirectional model - each intermediate frame depends on a mixture of not only the past, but future frames as well. Meanwhile, in our approach each frame only as a probabilistic dependence on the past frame through the entropy model, as we intend our approach to eventually be applied to an online setting.

2.2 Analysis of VTL: High-Frequency Information

When we initially view the rate-distortion curves in Fig. 3, we see that our approach underperforms video codecs by a large margin, especially on PSNR. In order to understand the quantitative discrepancy, we qualitatively analyze the frames of the source video and reconstructed video, as shown in Fig. 4. We observe that there exist high-frequency information in the source frames, often in the form of artifacts (see the color bands across the bridge), that our frame encoder is not able to capture even at a reasonably high bitrate.

We offer some hypotheses and discussions of these results. In our approach, each frame is encoded and reconstructed independently with an image encoder

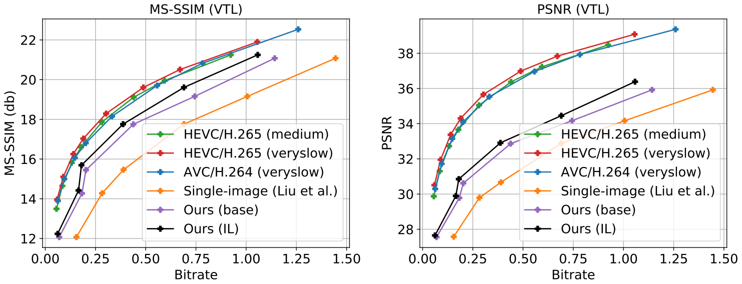

Fig. 3: Rate-distortion plot of our approach vs. competing baselines on VTL.

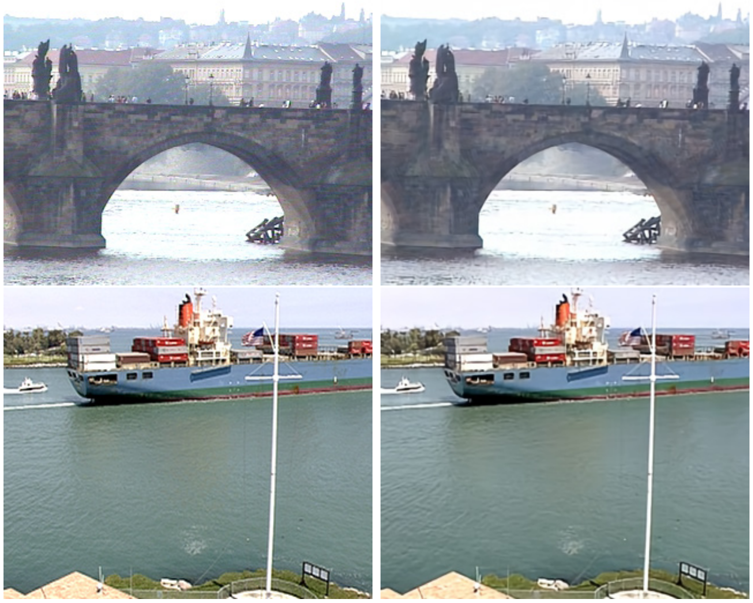

Fig. 4: Qualitative comparison of source VTL frames (left) vs. our reconstructions (right)

/ decoder. Our encoder/decoder may contain an inductive bias that might not be able to capture the high-frequency artifacts in the full frame, which explains why they are essentially “denoised” in the reconstruction. This could be due to the architecture of the encoder/decoder, or due to training the model on a different source dataset (Kinetics). And because our conditional entropy model only reduces the entropy/bitrate and doesn’t improve reconstruction in the same way motion estimation/interpolation does, the performance of our approach is only dependent on the quality of the reconstructions provided by the image encoder/decoder. In the meantime, video codecs utilizing explicit transformations only have to encode full frame information in the I-frames, leaving P-frames and B-frames to only encode the residual high-frequency artifact information.

There are pros and cons for both approaches. Explicit transformations may reconstruct high-frequency information better in intermediate frames, but can introduce additional artifacts through erroneous/quantized motion estimation and residual coding. Our approach of independent frame encoding guarantees no additional motion artifacts will be introduced, but how well high-frequency information can be encoded depends on the nature of the frame encoder.

2.3 Requesting Access to JCT-VC Test Sequences

We were not able to obtain access to the JCT-VC [1] test sequences for evaluation. We received the following response from the JCT-VC chairs: “Due to copyright restrictions, the JCT-VC and JVET databases of test sequences are only available to accredited participants in our image/video standardization activities. The contributors of some of those test sequences do not allow us to give access to everyone, and we need to respect the decisions of the copyright holders...The JCT-VC and JVET databases of test sequences are not intended for general use or for academic research purposes.”

3 Additional Runtime Details

We provide additional runtime details below. Specifically, we provide 1) additional comparisons of the entropy coding implementation of prior works, where made available by the authors, and 2) the runtime of our video codec baselines. The results are shown below, in Tab. 1, 2:

ms / frame	Wu et al. [7]	Lu et al. [4]	Ours
Encoding	281	$\sim$ 9000	140
Decoding	$\sim$ 800000 (joint with GPU decoding)	$\sim$ 9000	139

Table 1: Comparison of entropy coding implementation runtime among prior work.

Our entropy coding implementation is faster than those of prior work. Furthermore, our approach is on par with H.265 veryslow in terms of encoding time,

	H.265 veryslow	H.265 medium	H.264 veryslow	HEVC HM	AVC JM
ms / frame	914	93	220	$\sim$ 24000	$\sim$ 8400

Table 2: Comparison of runtime among video codec benchmarks.

and is significantly faster than HEVC HM and AVC JM. In general HEVC HM and AVC JM are orders of magnitude slower than their ffmpeg libx265/libx264 (H.265/H.264) counterparts. As always, we note that there are numerous opportunities to optimize our implementation further, as the bulk of our end-to-end runtime cost (1.19 s for encoding, 0.65 s for decoding) is due to I/O between CPU/GPU/the file system.

4 Additional Architecture Details

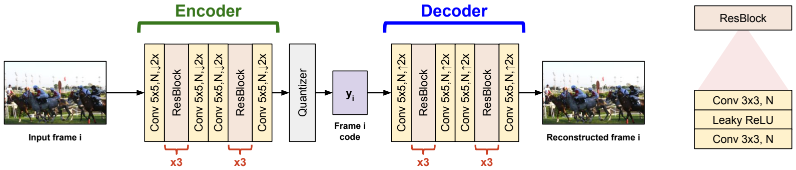

Fig. 5: Diagram of our image encoder and decoder model.

4.1 Architecture Details of Image Encoder / Decoder

The architecture of our image encoder and decoder are inspired from that of [3]. A diagram is shown in Fig. 5. The encoder consists of 5×5 downsampling convolutional layers as well as residual blocks in between. Each residual block consists of a simple sequence of 3×3 conv, Leaky ReLU, 3×3 conv layer. The decoder consists of a similar architecture, except each downsampling conv in the encoder is now replaced with a 5×5 upsampling transposed convolution in the decoder. Let the number of channels in each conv layer be denoted as N (this includes the number of channels of the quantized code). For the lower bitrates, we set $N = 192$, and for the higher bitrates, we set $N \in [250, 350]$. Since our conditional entropy model imposes its own information bottleneck depending on the λ we set in the rate-distortion loss function as well as the target bitrate R_a we want to enforce (see Section 3.3), we intentionally set the number of channels to be higher than necessary so that the channel dimensions themselves do not unnecessarily constrain information. We have not attempted to tune the number of channels for speed performance, though that would certainly provide further gains in speed.

As mentioned in Section 3.1 in the main paper, we note that we removed all non-local layers as they imposed a large bottleneck of speed and memory usage.

4.2 Additional Architecture Details of Hyperprior Encoder / Decoder

The architecture of the hyperprior encoder and decoder is listed in Fig. 4 in the main paper. Each residual block referred to in that figure is the same as the residual block defined above in Section 4.1 and Fig. 5 here in supplementary material.

In the hyperprior decoder, all feature maps at the spatial resolution of the main image code $\boldsymbol{y}_i$ or lower have N channels, with N being the same as the one defined above. All feature maps at a higher spatial resolution (the ones interspersed with the upsampling/downsampling IGDN/GDN layers) have M channels, with the exception of the highest resolution channel (which has 5 channels). M ranges from 80 at lower bitrates to 192 at higher bitrates. Each convolution layer in the hyperprior decoder originally has a kernel size of 5×5, though we replace the layer with two 3×3 conv layers for speed gains.

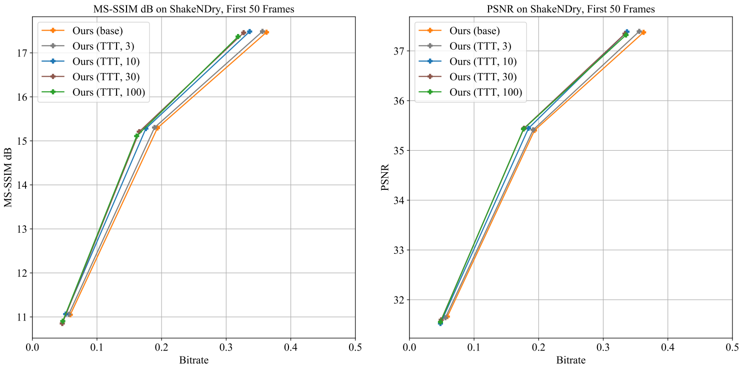

Fig. 6: MS-SSIM and PSNR rate-distortion plot on the first 50 frames of the UVG ShakeNDry video sequence of our approach as we vary the number of internal learning steps.

5 Internal Learning - Effect of Num. Steps on the Rate-distortion Curve

We additionally analyze how much the performance of our models improve as we increase the number of gradient steps used in internal learning of our frame latent codes $\boldsymbol{y}_i$ and $\boldsymbol{z}_i$ for each frame i. To do this, we evaluate on the first 50 frames of the `ShakeNDry` UVG video sequence.

Results are shown in Fig. 6 for both MS-SSIM and PSNR, where we plot the rate-distortion curves of our base model as well as with internal learning of 1, 10, 30, 100 gradient steps. Specifically, we use stochastic gradient descent with Nesterov momentum of 0.9 - we decrease the learning rate at higher gradient steps to reduce instability. We can see that in general, increasing the number of gradient steps reduces bitrate and distortion, though performance appears to saturate after 30 gradient steps. In the main paper, we use a fixed number of 10 steps for every test video frame.

6 Effect of GoP Size on Codec Performance

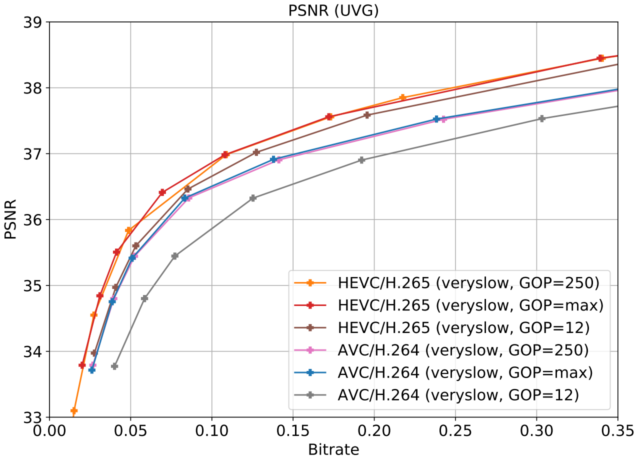

Fig. 7: Analysis of GoP size on codec performance in UVG.

We benchmark H.265 and H.264 on different Group of Pictures (GoP) settings to analyze how GoP affects codec performance. We adjust the maximum and minimum GoP size by tuning `keyint/min-keyint` and `keyint/keyint_min` in libx265 and libx264 respectively, in `ffmpeg`. We test using the default `ffmpeg` settings: $(max = 250, min = 25)$, as well as $(max = 12, min = 1)$ and $(max = \text{length of video}, min = 1)$. The results over UVG video are plotted in Fig. 7. We see that there is a marginal gap between a smaller GoP size of 12 and the default settings; the gap between the default settings and the maximum GoP size is thin.

7 Effect of Number of Input Frames on Performance

Recall that the hyperprior decoder in our entropy model only relies on the previous frame $\boldsymbol{y}_{i-1}$ as well as the current hyperprior code as input to produce distributions for the current code $\boldsymbol{y}_i$, due to the fact that we make a 1st-order Markov assumption. This was done intentionally to preserve the simplicity and efficiency of our model. However, for the sake of comparison, we decided to vary k as the number of past frames input into our hyperprior encoder/decoder to expand beyond our first-order assumption. We experiment with a frame encoder/decoder pre-trained on NorthAmerica and UVG, hence fixing the reconstruction quality. We vary k with a simple concatenation baseline - concatenating the k past frames as one tensor and slightly modifying the architecture of the hyperprior encoder/decoder to accommodate this input. The results are as follows:

k	1	2	3	4
Bitrate (NorthAmerica)	0.381	0.378	0.375	0.378
Bitrate (UVG)	0.175	0.186	0.179	0.182

Table 3: Concatenation of multiple past frames (k) as input to our entropy model, for both NorthAmerica/UVG. Reconstruction quality is fixed.

Given these results, we observe that it is non-trivial to reduce the bitrate of videos simply by concatenating additional past inputs into our entropy model.

References

1. Jct-vc: Common test conditions and software reference configurations, available at https://hevc.hhi.fraunhofer.de/. Accessed 2020/03/01.
2. Video trace library, available at http://trace.kom.aau.dk/. Accessed 2019/11/01.
3. Liu, H., Chen, T., Guo, P., Shen, Q., Cao, X., Wang, Y., Ma, Z.: Non-local Attention Optimized Deep Image Compression. ArXiv (2019)
4. Lu, G., Ouyang, W., Xu, D., Zhang, X., Cai, C., Gao, Z.: Dvc: An end-to-end deep video compression framework. CVPR (2019)
5. Townsend, J., Bird, T., Barber, D.: Neural inter-frame compression for video coding. ICCV (2019)
6. Wang, H., Gan, W., Hu, S., Lin, J.Y., Jin, L., Song, L., Wang, P., Katsavounidis, I., Aaron, A., Kuo, C.C.J.: Mcl-jcv: A jnd-based h.264/avc video quality assessment dataset. ICIP (2016)
7. Wu, C.Y., Singhal, N., Krahenbuhl, P.: Video compression through image interpolation. ECCV (2018)