Feedback Recurrent Autoencoder for Video Compression
Supplementary materials

A Model, training and datasets details

A.1 Datasets

Our training dataset was based on Kinetics400 [38]. We selected a subset of the high-quality videos in Kinetics (width and height over 720 pixels), took the first 16 frames of each video, and downscaled them (to remove compression artifacts) such that the smaller of the dimensions would be 256 pixels. At train time, random crops of size 160×160 were used. For validation, we used videos from HDgreetings, NTIA/ITS and SVT from Xiph.org [39].

We used UVG [14] and HEVC Classes BCDE [15] as our test datasets. UVG contains 7 1080p video sequences and a total of 3900 frames. Both the UVG and HEVC BCDE sequences are available in YUV420-8bit format. UVG 1080p frames were converted to RGB using OpenCV [47]. HEVC sequences vary in resolution per class and use `ffmpeg` [41] for RGB conversion.

A.2 Training

All implementations were done in the PyTorch framework [48]. The distortion measure $1-$MS-SSIM [16] was normalized per frame and the rate term was normalized per pixel (i.e. bits per pixel) before they were combined $D + \beta R$. We trained our network with five β values $\beta = \{0.025, 0.05, 0.1, 0.2, 0.3\}$ to obtain a rate-distortion curve. The I-frame network and P-frame network were trained jointly from scratch without pretraining for any part of the network.

The GoP size of 8 was used during training, i.e. the recurrent P-frame network was unrolled 7 times. We used a batch size of 16, for a total of 250k gradient updates (iterations) corresponding to 20 epochs. When training with the flow enhancement network MENet, we applied $\mathcal{L}_{\text{fe}}$ and $\mathcal{L}_{\text{fd}}$ until 20k iterations, and the transition of the input from $\mathbf{x}_{t-1}$ to $\widehat{\mathbf{x}}_{t-1}$ was done at iteration 15k. All models were trained using Adam optimizer [49] with the initial learning rate of 10^{-4}, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate was annealed by a factor of 0.8 every 100k iterations. Performance on the validation set was evaluated every epoch, results presented below are the model checkpoints performing best on the validation set. See Appendix A for details on the architecture and training.

A.3 Model

Detailed architecture of the encoder and decoder in Fig. 2(a) is illustrated in Fig. 9. For prior modeling, we use the same network architecture as described in Section B.2 of [6]. Detailed architecture of the optical flow estimation network in Fig. 2(b) is illustrated in Fig. 10.

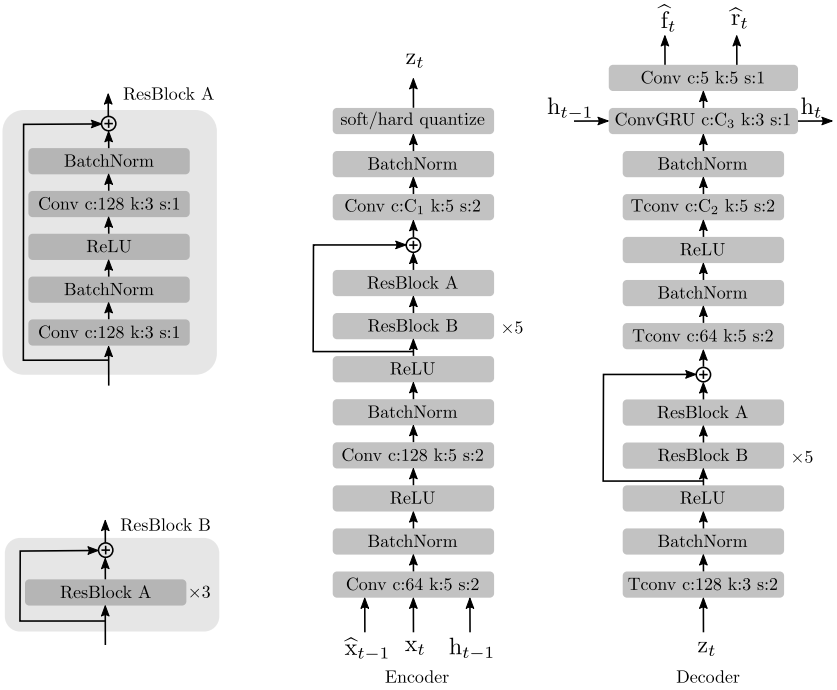

Fig. 9: Architecture of our P-frame network (flow enhanced module is separately illustrated). *Tconv* denotes transposed convolution. For (transposed) convolutional layer c denotes the number of output channels, k denotes the kernel size and s denotes the stride. In all of our experiments, we set $C_1 = 32$, $C_2 = 10$, and $C_3 = 32$. Architecture of I-frame network differs by removing the final two layers in the decoder, setting $C_2 = 3$, and having $\mathbf{x}_0$ as the only input to the encoder. Note that a large part of this architecture is inherited from Fig. 2 in [25] and Fig. 9 in [6].

As can be seen from Fig. 9, we use BatchNorm throughout the encoder and decoder architecture to help stabilize training. However, since previous decoded frame is fed back into the encoder in the next time step, during training it is not possible to apply the same normalization across all the unrolled time steps, which will lead to an inconsistency across training and evaluation. To solve this issue, we switch BatchNorm to evaluation mode during training at iteration 40k, after which its parameters are no longer updated.

B Images used in figures

Video used in Figures 7, 6, 13, 12, 15, and 16 is produced by Netflix, with `CC BY-NC-ND 4.0` license:
`https://media.xiph.org/video/derf/ElFuente/Netflix_Tango_Copyright.txt`

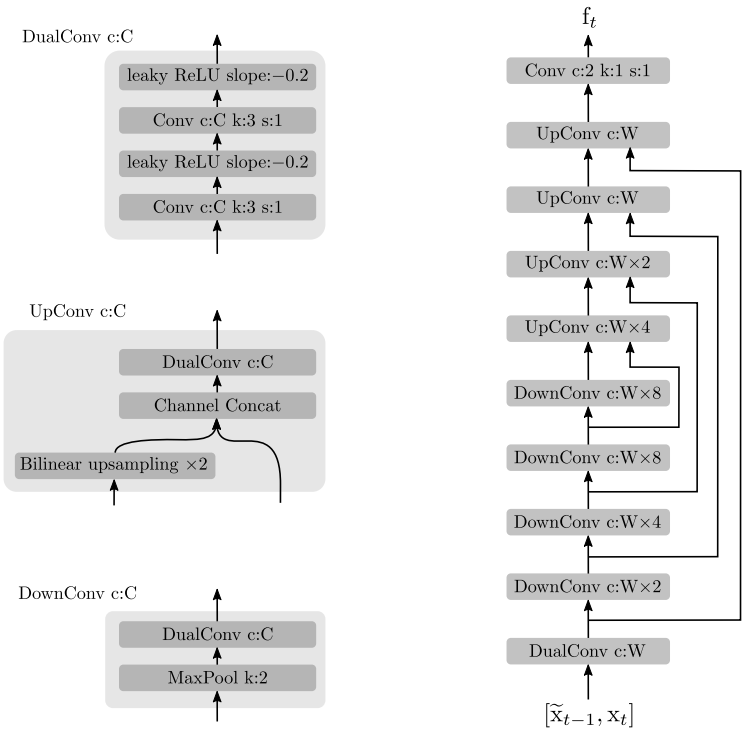

Fig. 10: Architecture of our optical flow estimation network. In all of our experiments, we set $W = 16$.

C `ffmpeg` and `HM` experiments

We generated H.264 and H.265 baselines using `ffmpeg` and `HM` software tools. Both `HM` and `ffmpeg` received the UVG videos in the native YUV-1080p-8bit format as inputs. The command that we used to run `ffmpeg` in low-latency mode is as follows:

```
ffmpeg -y -pix_fmt yuv420p -s [W]x[H] -r [FR] -i [IN].yuv
-c:v libx[ENC] -b:v [RATE]M -maxrate [RATE]M -tune zerolatency
-x[ENC]-params "keyint=[GOP]:min-keyint=[GOP]:verbose=1" [OUT].mkv
```

and the command that we used to run `ffmpeg` in default settings is as follows:

```
ffmpeg -y -pix_fmt yuv420p -s [W]x[H] -r [FR] -i [IN].yuv
-c:v libx[ENC] -b:v [RATE]M -maxrate [RATE]M
-x[ENC]-params "verbose=1" [OUT].mkv
```

where the values in brackets represent the encoder parameters as follows: `H` and `W` are the frame dimensions (1080×1920), `FR` is the frame rate (120), `ENC` is the encoder type (x264 or x265), `GOP` is the GoP size (12 for low-latency settings), `INPUT` and `OUTPUT` are the input and the output filenames, respectively, `RATE` controls the intended bit rate in Mega-bits/second (We tried `RATE`=

$\{10, 20, 37, 62, 87, 112\}$ Mega-bits/second that translate to $\{0.04, 0.08, 0.15, 0.25, 0.35, 0.45\}$ bits/pixel for UVG 1080p at 120 fps). It is worth to mention that we did not set the `-preset` flag in `ffmpeg` that controls the coding performance. As a result, it used the default value which is `medium`, unlike some existing papers that employed `ffmpeg` in `fast` or `superfast` presets that led to poor `ffmpeg` performance.

The command we used to run `HM` is as follows:

```
TAppEncoderStatic -c [CONFIG].cfg -i INPUT.yuv -wdt [W] -hgt [H]
-fr [FR] -f [LEN] -o OUTPUT.yuv -b -ip [GOP] -q [QP] > [LOG].log
```

where `CONFIG` is the `HM` configuration file (we used `LowDelayP.cfg`), `LEN` is the number of frames in the video sequence (600 or 300 for UVG videos), `LOG` is the log file name that we used to read the bit rate values, and `QP` control the quality of the encoded video that leads to different bit rates (we tried `QP`$=\{20, 22, 25, 30, 35, 40\}$ in this work).

MS-SSIM for all the experiments was calculated in RGB domain in video-level. bit rates for `ffmpeg` were calculated by reading the compressed file size and for `HM` by parsing the `HM` log file. Both values were then normalized by frame-dimensions and frame-rate.

C.1 YUV to RGB conversion inconsistencies

Since the UVG raw frames are available in YUV format and our model works in RGB domain only, we had to convert the UVG frames from YUV to RGB color space. We considered two methods to convert UVG-1080p-8bit videos to RGB 1080*p*, *i)* use `ffmpeg` to read in YUV format and directly save the frames in RGB format, *ii)* read the frames in YUV and convert them to RGB via `COLOR_YUV2BGR_I420` functionality provided by the `OpenCV` [47] package. We also tried a third scenario, *iii)* use `ffmpeg` to read UVG-4K-10bit in YUV format and save the RGB 1080*p* frames. Fig. 11 shows the performance of our model on the three different versions of UVG RGB. As can be seen from this figure, there is an inconsistency in our model performance across different UVG versions. A potential cause for this behavior could be the interpolation technique employed in the color conversion method, since in YUV420 format U and V channels are subsampled and need to be upsampled before YUV to RGB conversion. The results reported in this paper are based on the `OpenCV` color conversion.

D Qualitative examples

We provide in this section more qualitative examples for our method in Fig. 12 and for H.265 in Fig. 13. Notably, we overlay the MS-SSIM and MSE error maps on the original frame to provide indication of the distortion characteristics of each algorithm. In addition in Fig. 12, we show the BPP maps, as we have access to the entropy of the corresponding latent using the I-frame and P-frame prior

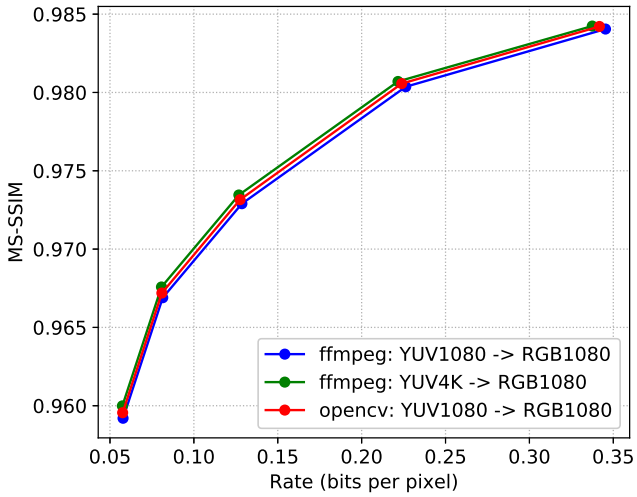

Fig. 11: Effect of conversion method on model performance on UVG dataset.

models. We average the entropy across channels, then use bilinear upsampling to bring the BPP map back to the original image resolution.

By examining the BPP maps in Fig. 12 (third row), we can see that intuitively the model spends bits more uniformly across I-frames (first and last column), while during P-frames bits are mostly spent around edges of moving objects.

Note that our method optimizes for MS-SSIM, while traditional video codecs like H.265 are evaluated with PSNR. As detailed in Appendix G, MS-SSIM is insensitive to uniform color shifts and mainly focuses on spatial structure. On the contrary, PSNR does pick up color shifts, but it tends to care less for fine details in spatial structure, i.e. texture, sharp edges. Our method's MS-SSIM maps (Fig. 12 4th row) show edges are sharply reconstructed, yet does not fully capture some of the color shifts in the reconstruction. In particular the man in the pink suit has a consistently dimmer pink suit in the reconstruction (compare first and second row), which does not appear in the MS-SSIM loss, while MSE seems to pick up the shift. In contrast in Fig. 13, H.265 MSE maps (last row) shows that the error is concentrated around most edges and fine textures due to blurriness, yet most colors are on target.

We observe that for our method, MS-SSIM error maps degrade as we go from I-frame to P-frame (compare first and second to last column in Fig. 12). Similarly, the BPP map of the first I-frame show a large rate expenditure, while consecutive P-frames display significantly less rate. Both of these qualitative results confirm the quantitative analysis of Fig. 8.

Fig. 12: Our method's qualitative evaluation on frames from Netflix Tango in Netflix El Fuente. The rows show the original frames, the reconstructed frames, then the BPP, MS-SSSIM and MSE maps overlayed on top of the grayscale original frames. Each column shows a different frame from a full cycle of GoP 8 going from I-frame to I-frame, showing every other frame.

Fig. 13: H.265 qualitative evaluation on frames from Netflix Tango in Netflix El Fuente. The rows show the original frames, the reconstructed frames, then the MS-SSSIM and MSE maps overlayed on top of the grayscale original frames. Each column shows a different frame from a full cycle of GoP 8 going from I-frame to I-frame, showing every other frame.

E Theoretical justification of the feedback recurrent module

In this subsection we provide theoretical motivations on having feedback recurrency module in our network design. Specifically, we show that, for the compression of sequential data with a causal decoder and causal encoder, (i) it is imperative for encoder to have a memory that summarizes previously latent codes, and (ii) it is not beneficial for encoder to access history input information.

Below is an abstract view of a generic sequential autoencoder, where $\mathbf{X} = \{\mathbf{x}_\tau\}_{\tau\in\mathbb{N}}$ denotes a discrete-time random process that represents the empirical distribution of our training video data, with $\mathbf{Z}$ and $\widehat{\mathbf{X}}$ being the induced latent and reconstruction processes.

$$\mathbf{X} \xrightarrow{f_{\text{enc}}} \mathbf{Z} \xrightarrow{f_{\text{dec}}} \widehat{\mathbf{X}}.$$

From an information theoretical point of view, the sequential autoencoding problem can be abstracted as the maximization of average frame-wise mutual information between $\mathbf{X}$ and $\widehat{\mathbf{X}}$, detailed below, with certain rate constraint on $H(\mathbf{Z})$,

$$\begin{aligned} &\max_{f_{\text{enc}},f_{\text{dec}}} \sum_\tau I\left(\mathbf{x}_\tau;\widehat{\mathbf{x}}_\tau\right), && \text{frame-wise mutual information}^{6} \\ &\text{s.t. } I\left(\mathbf{x}_\tau;\widehat{\mathbf{x}}_\tau|\mathbf{z}_{\le\tau}\right)=0,\ \forall\tau. && \text{decoder causality} \end{aligned}$$

The decoder causality is encoded in such a form as $I(\mathbf{x}_\tau;\widehat{\mathbf{x}}_\tau\ |\mathbf{z}_{\le\tau})$ is zero *if and only if* $\widehat{\mathbf{x}}_\tau$ is not a function of $\mathbf{z}_{>\tau}$. It is important to note that mutual information is invariant to any bijections and thus would not reflect perceptual quality of reconstruction. Nevertheless, we use this formulation only to figure out important data dependencies from an information theory perspective and use that to guide us in the design of network architecture. With a closer look, the t^{th} term in the objective function can be rewritten as below.

$$\begin{aligned} & I(\mathbf{x}_t;\widehat{\mathbf{x}}_t) \\ =& I(\mathbf{x}_t;\widehat{\mathbf{x}}_t,\mathbf{z}_{\le t}) - I(\mathbf{x}_t;\mathbf{z}_{\le t}|\widehat{\mathbf{x}}_t) \\ =& I(\mathbf{x}_t;\mathbf{z}_{\le t}) + I(\mathbf{x}_t;\widehat{\mathbf{x}}_t|\mathbf{z}_{\le t}) - I(\mathbf{x}_t;\mathbf{z}_{\le t}|\widehat{\mathbf{x}}_t) \\ (a)\ =& I(\mathbf{x}_t;\mathbf{z}_{\le t}) - I(\mathbf{x}_t;\mathbf{z}_{\le t}|\widehat{\mathbf{x}}_t) \\ =& I(\mathbf{x}_t;\mathbf{z}_{<t}) + I(\mathbf{x}_t;\mathbf{z}_t|\mathbf{z}_{<t}) - (I(\mathbf{x}_t;\mathbf{z}_{<t}|\widehat{\mathbf{x}}_t) + I(\mathbf{x}_t;\mathbf{z}_t|\widehat{\mathbf{x}}_t,\mathbf{z}_{<t})) \\ (b)\ =& \underbrace{I(\mathbf{x}_t;\mathbf{z}_{<t};\widehat{\mathbf{x}}_t)}_{\text{prediction}} + \underbrace{I(\mathbf{x}_t;\mathbf{z}_t;\widehat{\mathbf{x}}_t|\mathbf{z}_{<t})}_{\text{innovation}}. \end{aligned} \tag{3}$$

[6] Note that this is different from $I(\mathbf{X};\widehat{\mathbf{X}})$. If we instead maximize $I(\mathbf{X};\widehat{\mathbf{X}})$, then we could have a single output frame capturing the information of more than one input frames, which is not what we want.

Step (a) incorporates the decoder causality constraint, step (b) comes from the definition of multi-variate mutual information[7], and the rest uses the identity of conditional mutual information[8].

Equation (3) says that $I(\mathbf{x}_t; \widehat{\mathbf{x}}_t)$ can be broken down into two terms, the first represents the prediction of $\mathbf{x}_t$ from all previous latents $\mathbf{z}_{<t}$, and the second represents new information in $\mathbf{z}_t$ about $\mathbf{x}_t$ that is not be explained by $\mathbf{z}_{<t}$. While this formulation does not indicate how the optimization can be done, it tells us what variables should be incorporated in the design of f_{enc}. Let us focus on $\mathbf{z}_t$: in our objective function $\sum_\tau I(\mathbf{x}_\tau; \widehat{\mathbf{x}}_\tau)$, $\mathbf{z}_t$ shows up in the following terms

$$I(\mathbf{x}_\tau; \mathbf{z}_\tau; \widehat{\mathbf{x}}_\tau | \mathbf{z}_{<\tau}) \text{ for } \tau \geq t, \text{ and } I(\mathbf{x}_\tau; \mathbf{z}_{<\tau}; \widehat{\mathbf{x}}_\tau) \text{ for } \tau > t.$$

It is clear, then, that the optimal $\mathbf{z}_t$ should only be a function of $\mathbf{x}_{\geq t}$, $\widehat{\mathbf{x}}_{\geq t}$, $\mathbf{z}_{<t}$ and $\mathbf{z}_{>t}$. Combining it with the constraint that the encoder is causal, we can further limit the dependency to $\mathbf{x}_t$ and $\mathbf{z}_{<t}$. In other words, it suffices to parameterize the encoder function as $\mathbf{z}_t = f_{\text{enc}}(\mathbf{x}_t, \mathbf{z}_{<t})$, which attests to the two claims at the beginning of this subsection: (i) $\mathbf{z}_t$ should be a function of $\mathbf{z}_{<t}$ and (ii) $\mathbf{z}_t$ does not need to depend on $\mathbf{x}_{<t}$.

It is with these two claims that we designed the network where the decoder recurrent state, which provides a summary of $\mathbf{z}_{<t}$, is fed back[9] to the encoder as input at time step t and there is no additional input related to $\mathbf{x}_{<t}$.

It is worth noting that in [28,29], the authors introduce a neural network architecture for progressive coding of images, and in it an *encoder recurrent connection* is added on top of the feedback recurrency connection from the decoder. Based on the analysis in this section, since the optimization of $\mathbf{z}_t$ does not depend on $\mathbf{x}_{<t}$, we do not include encoder recurrency in our network design.

F Graphical modeling considerations

In this section we give more details on the reasoning behind how we formulated the graphical models presented in Fig. 3.

Lu *et al.*, DVC [2,3]
Generative model. Temporally independent prior is used hence no edges between the latent variables $\mathbf{z}_t$. Consecutive reconstructions $\widehat{\mathbf{x}}_t$ depend on the previous reconstructions $\widehat{\mathbf{x}}_{t-1}$ and hence the edge between them $\widehat{\mathbf{x}}_{t-1} \to \widehat{\mathbf{x}}_t$.
Inference model. During inference, at timestep t, the current timestep's latent $\mathbf{z}_t$ is inferred based on the previous reconstruction $\widehat{\mathbf{x}}_{t-1}$ and current original frame $\mathbf{x}_t$. In turn, the previous reconstruction $\widehat{\mathbf{x}}_{t-1}$ is determined based on the information from the previous timestep's latent $\mathbf{z}_{t-1}$ and the earlier reconstruction

[7] $I(a; b; c) = I(a; b) - I(a; b|c)$. For a Markov process $a \to b \to c$, this indicates the amount of information flown from a to c through b.
[8] $I(a; b) = I(a; b, c) - I(a; c|b)$ for any c.
[9] Since $\widehat{x}_{<t}$ is a deterministic function of $\mathbf{z}_{<t}$, any additional input of $\widehat{x}_{<t}$ to the encoder is also justified.

$\widehat{\mathbf{x}}_{t-2}$. Since the procedure of determining $\widehat{\mathbf{x}}_{t-1}$ from $\mathbf{z}_{t-1}$ and $\widehat{\mathbf{x}}_{t-2}$ is deterministic, the inference model has an edge $\mathbf{z}_{t-1} \to \mathbf{z}_t$. Extending this point, since $\widehat{\mathbf{x}}_t$ is updated at each timestep using a deterministic procedure using its previous value $\widehat{\mathbf{x}}_{t-1}$ and $\mathbf{z}_t$, it makes $\mathbf{z}_t$ dependent on all the past latent codes $\mathbf{z}_{<t}$. Hence there are edges from all the past latents $\mathbf{z}_{<t}$ to the present latent $\mathbf{z}_t$.
The edge $\mathbf{x}_t \to \mathbf{z}_t$ arises from the direct dependence of $\mathbf{z}_t$ on $\mathbf{x}_t$ through the encoder.

Liu *et al.* [7]
Generative model. Due to the introduction of the time-autoregressive code prior there are additional edges $\mathbf{z}_{<t} \to \mathbf{z}_t$ as compared to model (a) with dashed edges.
Inference model. The inference model remains unchanged.

Lin *et al.*, M-LVC [9]
Generative model. Due to the introduction of warping referencing $k > 1$ previously reconstructed frames $\widehat{\mathbf{x}}_{t-k:t-1}$ for the purpose of reconstructing $\widehat{\mathbf{x}}_t$ there are additional edges $\widehat{\mathbf{x}}_{t-k:t-2} \to \widehat{\mathbf{x}}_t$ as compared to DVC, i.e. model (a) with solid lines only. Also, previously reconstructed flow values $\widehat{v}_{t-k:t-1}$ (as per notation of [9]) are utilized for the purpose of warping $\widehat{\mathbf{x}}_{t-k:t-1}$ what motivates additional edges $\mathbf{z}_{t-k+1:t-1} \to \widehat{\mathbf{x}}_t$. Fig. 3(a), including all the solid and dotted lines, presents the graphical model for the buffer of size $k = 2$ previously reconstructed frames. Note that there is no edge $\mathbf{z}_1 \to \widehat{\mathbf{x}}_2$ since the I-frame doesn't carry any flow information.
Inference model. The inference model remains unchanged.

Rippel *et al.* [5] and ours
Generative model. Model (b) differs from model (a) in the introduction of the recurrent connection and hence hidden state $\mathbf{h}_t$, hence the full graphical model could be drawn as per Fig. 14. However, the $\mathbf{h}_t$ is not a stochastic random variable, but a result of a deterministic function of $\mathbf{h}_{t-1}$ and $\mathbf{z}_t$. For that reason we can drop explicitly drawing nodes $\mathbf{h}_t$ and move the deterministic relationship they are implying into the graphical model edges between the latent variables $\mathbf{z}_t$. This way we arrive from the models in Fig. 14 to models in Fig. 3(b).
Inference model. Analogous reasoning applies in the case of the inference model.

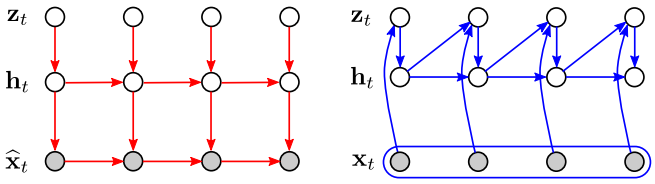

Fig. 14: Left: Generative model of our method with intermediate variable $\mathbf{h}_t$. Right: Inference model of our method with intermediate variable $\mathbf{h}_t$.

Han *et al.* [8]
Graphical models are derived based on equations by Han *et al.* [8]: Eq. (4) and Eq. (6) for the generative model, and Eq. (5) for the inference model.

Habiban *et al.* [6]
Generative model. Due to the use of a prior which is autoregressive in the temporal dimension (across the blocks of 8 frames that Habiban *et al.* are using) we draw edges $\mathbf{z}_{<t} \rightarrow \mathbf{z}_t$. The latent codes are generated by a deterministic composition of 3D convolutional layers with the input of the original frames. The temporal receptive field is larger than original input sequence which means that every latent variable in the block can be influenced by each of the original input frames and hence the fully connected clique between $\mathbf{z}_{1:T}$ and $\mathbf{x}_{1:T}$.
Inference model. The same reasoning applies for the inference model.

F.1 Full flexibility of the marginal $\mathbb{P}_{\mathbf{X}}(\mathbf{x}_{1:T})$

The graphical model corresponding to the approach of Liu *et al.* [7] is presented in Fig. 3(a) including dashed lines. In this case, the marginal $\mathbb{P}_{\mathbf{X}}(\mathbf{x}_{1:T})$ is fully flexible in the sense that it does not make any conditional independence assumptions between $\mathbf{x}_{1:T}$.

To see that consider considering what happens when we marginalize out $\mathbf{z}_{1:T}$ from $\mathbb{P}_{\mathbf{Z}}(\mathbf{z}_{1:T})\mathbb{P}_{\mathbf{X}|\mathbf{Z}}(\mathbf{x}_{1:T}|\mathbf{z}_{1:T})$ using the variable elimination algorithm [35]. Assume we use an elimination order τ which is permutation of a set of numbers $\{1, \ldots, T\}$, e.g., for $T = 4$ a viable permutation is $\tau = \{2, 1, 4, 3\}$. Let's think about the process of variable elimination in terms of the induced graph. Eliminating the first variable $\mathbf{z}_{\tau_1}$ induces a connection between $\mathbf{x}_{\tau_1}$ and each of the other latent variables $\mathbf{z}_{\neq\tau_1}$. Eliminating consecutive latent variables $\mathbf{z}_t$ will induce connections between corresponding observed variables $\mathbf{x}_t$ and every other $\mathbf{z}_{\neq t}$. This way the final latent variable to be eliminated $\mathbf{z}_{\tau_T}$ will be connected in the induced graph with all the observed variables $\mathbf{x}_{1:T}$. Hence when we eliminate $\mathbf{z}_{\tau_T}$ that will result in a factor $\phi(\mathbf{x}_{1:T})$ (a clique between all nodes $\mathbf{x}_{1:T}$ thinking in terms 2of the induced graph). The marginal distribution we are looking for is $\mathbb{P}_{\mathbf{X}}(\mathbf{x}_{1:T}) \propto \phi(\mathbf{x}_{1:T})$, and it makes no conditional independence assumptions.

The graphical model corresponding to our and Rippel *et al.* approach is presented in Fig. 3(b). In this case, showing that the marginal $\mathbb{P}_{\mathbf{X}}(\mathbf{x}_{1:T})$ makes no conditional independence assumptions between $\mathbf{x}_{1:T}$ is simpler – no matter what variable elimination order we choose, since $\mathbf{z}_1$ is connected to all the nodes $\mathbf{x}_{1:T}$ eliminating $\mathbf{z}_1$ will always result in a factor $\phi(\mathbf{x}_{1:T})$.

G MS-SSIM color issue and corner artifact

G.1 MS-SSIM color shift artifact

MS-SSIM, introduced in [16], is known to be to some extent invariant to color shift [46]. Inspired by the study in Fig. 4 of [50], we designed a small experiment to study the extent of this issue. We disturbed an image $\mathbf{x}$ slightly with additive

Gaussian noise to obtain $\widehat{\mathbf{x}}_0$, such that $\widehat{\mathbf{x}}_0$ lies on a certain equal-MS-SSIM hypersphere (we choose 0.966 which corresponds to the average quality of our second lowest rate model). Then we optimize $\widehat{\mathbf{x}}$ for lower PSNR, under the constraint of equal MS-SSIM using the following objective:

$$\mathcal{L} = \text{PSNR}(\mathbf{x}, \hat{\mathbf{x}}) + \alpha_t \cdot |\text{MS-SSIM}(\mathbf{x}, \hat{\mathbf{x}}) - \text{MS-SSIM}(\mathbf{x}, \hat{\mathbf{x}}_0)| \tag{4}$$

We use Adam optimizer with a learning rate of 10^{-4} (otherwise default parameters) for $T = 20,000$ iterations. We use $\alpha_t = \min(1, \frac{2t}{T}) \times 10^3$, with t representing the iteration index.

In Fig. 15, (a) corresponds to $\mathbf{x}$, (b) corresponds to $\hat{\mathbf{x}}_0$, (c) to $\hat{\mathbf{x}}$ at iteration $2,000$ and (d) to $\hat{\mathbf{x}}$ at iteration $20,000$. All frame (b), (c) and (d) are on the equal-MS-SSSIM hypersphere of 0.966 with respect to the original frame $\mathbf{x}$. We can observe that it is possible to obtain an image with very low PSNR, mainly due to drastic uniform color shifts, while remaining at equal distance in terms of MS-SSIM from the original image. This helps explain some of the color artifacts we have seen in the output of our models.

G.2 MS-SSIM corner artifact

The standard implementation of MS-SSIM, when used in $\mathcal{L}_{\text{RD}}$, causes corner artifacts as shown in Figure 16. Although the artifact is subtle and barely noticeable when looking at full-size video frames, it is worth investigation. We found the root cause in the convolution-based implementation of local mean calculations in SSIM function where a Gaussian kernel (normally of size 11) is convolved with the input image. The convolution is done without padding, as a result the contribution of the corner pixels to MS-SSIM is minimal as they fall at the tails of the Gaussian kernel. This fact is not problematic when MS-SSIM is used for evaluation, but can negatively affect training. Specifically in our case, where MS-SSIM was utilized in conjunction with rate to train the network, the network took advantage of this deficiency and assigned less bits to the corner pixels as they are less important to the distortion term. As a result, the corner pixels appeared blank in the decoded frames. We fixed this issue by padding, e.g., replicate, the image before convolving it with the Gaussian kernel. The artifact was eliminated as can be seen from Fig. 16.

It is worth to mention that we used the standard MS-SSIM implementation in all the trainings and evaluations in this paper for consistency with other works.

H HEVC & PSNR results

The comparison using PSNR metric on UVG dataset are in Fig. 17 and the results for HEVC Classes BCDE datasets using both MS-SSIM and PSNR metrics are in Fig. 18.

(a) Original (b) PSNR 22.0dB

(c) PSNR 12.7dB (d) PSNR 9.5dB

Fig. 15: An illustration of MS-SSIM color issue. All frames (b), (c) and (d) have an MS-SSIM of 0.966. (a): original frame, (b): perturbed image with slight additive gaussian noise, (c) and (d): images obtained by optimizing (b) for $\mathcal{L} = \text{PSNR}$ under equal MS-SSIM contraint.
Crop of frame 229 of Tango video from Netflix Tango in Netflix El Fuente; see Appendix B for license information.

Fig. 16: An illustration of the MS-SSIM corner artifact, best viewed on screen. Left: uncompressed frame, middle: corners when trained with default MS-SSIM implementation, right. corners when trained with replicate-padded MS-SSIM implementation.
Frame 10 of Tango video from Netflix Tango in Netflix El Fuente; see Appendix B for license information.

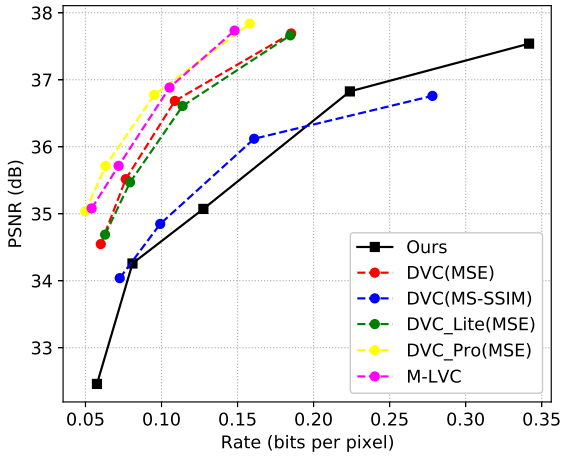

Fig. 17: Comparison to the state-of-the-art learned methods on the UVG dataset using PSNR distortion metric.

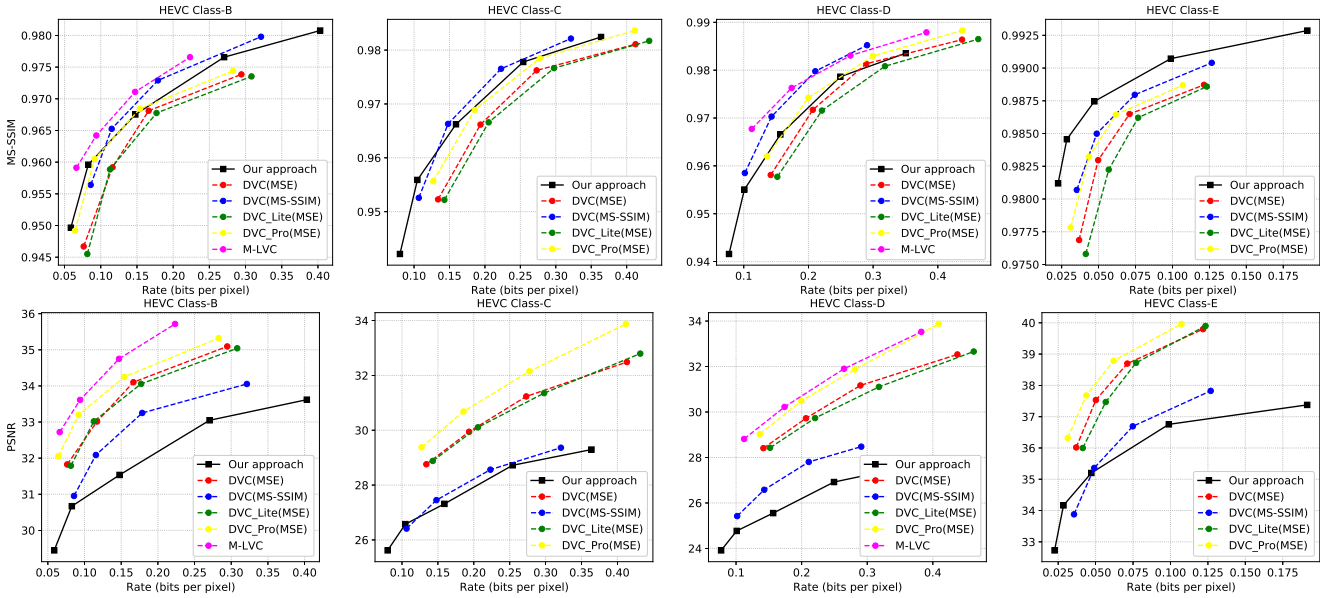

Fig. 18: Comparison to the state-of-the-art learned methods on HEVC Classes BCDE using MS-SSIM (top) and PSNR (bottom) distortion metrics.

I Computational efficiency

In its current form, the decoding speed is the bottleneck because of the use of an auto-regressive entropy model that factorizes over each latent dimension, which may hinder it from being directly applicable to real-time use-cases. Sidestepping this limitation without sacrificing the rate-distortion performance is a subject of ongoing research [51].

When comparing the computational performance of neural compression to HEVC one has to take into account that the performance of HEVC is a product of 30 years of algorithmic and hardware optimizations, e.g., our model is orders of magnitude faster in encoding and comparable in decoding w.r.t. HEVC reference implementation HM [42]. We anticipate similar hardware optimizations for CNNs and other neural architectures will yield similar acceleration for neural compression in the future.

J Generalization with respect to GoP size

In this section we take a look at the generalization of the model with respect to the GoP size. Fig. 19 shows the results achieved by a model trained with a GoP of 8 when evaluated with a GoP of 10, such that the recurrent model at runtime is unrolled for longer than at training time. Similarly as in Fig. 8, as expected, we see a gradual degradation of P-frame quality as it moves further away from the I-frame. However, in Fig. 19 we also see that the degradation for the frames index 8 and 9, i.e. the steps that were not unrolled at training time, accelerates significantly. The model in its current form and with the current method of training cannot dynamically adapt to larger GoP sizes.

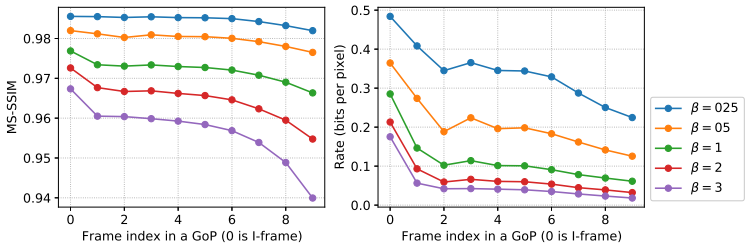

Fig. 19: Results achieved by a model trained with a GoP of 8 when evaluated with a GoP of 10.

References

1. Sandvine: 2019 Global Internet Phenomena Report. https://www.ncta.com/whats-new/report-where-does-the-majority-of-internet-traffic-come (2019) Accessed: 2020-02-28.
2. Lu, G., Ouyang, W., Xu, D., Zhang, X., Cai, C., Gao, Z.: DVC: An End-to-End Deep Video Compression Framework. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019)
3. Lu, G., Zhang, X., Ouyang, W., Chen, L., Gao, Z., Xu, D.: An end-to-end learning framework for video compression. IEEE Transactions on Pattern Analysis and Machine Intelligence (2020)
4. Wu, C.Y., Singhal, N., Krähenbühl, P.: Video Compression through Image Interpolation. In: Proceedings of the European Conference on Computer Vision. (2018)
5. Rippel, O., Nair, S., Lew, C., Branson, S., Anderson, A.G., Bourdev, L.: Learned Video Compression. In: Proceedings of the International Conference on Computer Vision. (2019)
6. Habibian, A., van Rozendaal, T., Tomczak, J.M., Cohen, T.S.: Video Compression With Rate-Distortion Autoencoders. In: Proceedings of the International Conference on Computer Vision. (2019)
7. Liu, H., shen, H., Huang, L., Lu, M., Chen, T., Ma, Z.: Learned Video Compression via Joint Spatial-Temporal Correlation Exploration. In: Proceedings of the national conference on Artificial Intelligence. (2020)
8. Han, J., Lombardo, S., Schroers, C., Mandt, S.: Deep Probabilistic Video Compression. In: Advances in Neural Information Processing Systems. (2019)
9. Lin, J., Liu, D., Li, H., Wu, F.: M-lvc: Multiple frames prediction for learned video compression. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2020)
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016)
11. Shi, X., Chen, Z., Wang, H., Yeung, D., Wong, W., Woo, W.: Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In: Advances in Neural Information Processing Systems. (2015)
12. van den Oord, A., Kalchbrenner, N., Espeholt, L., Kavukcuoglu, K., Vinyals, O., Graves, A.: Conditional Image Generation with PixelCNN Decoders. In: Advances in Neural Information Processing Systems. (2016)
13. Yang, Y., Sautiére, G., Ryu, J.J., Cohen, T.S.: Feedback Recurrent AutoEncoder. In: IEEE International Conference on Acoustics, Speech and Signal Processing. (2019)
14. Mercat, A., Viitanen, M., Vanne, J.: Uvg dataset: 50/120fps 4k sequences for video codec analysis and development. In: Proceedings of the 11th ACM Multimedia Systems Conference. MMSys '20, New York, NY, USA, Association for Computing Machinery (2020) 297–302
15. Bossen, F.: Common test conditions and software reference configurations. JCTVC-F900 (2011)
16. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P., et al.: Image quality assessment: from error visibility to structural similarity. IEEE Trans. on Image Processing **13** (2004) 600–612

17. Summerson, C.: How much data does Netflix use? https://www.howtogeek.com/338983/how-much-data-does-netflix-use/ (2018) Accessed: 2020-02-28.
18. Pearlman, W.A., Said, A.: Digital Signal Compression: Principles and Practice. Cambridge University Press (2011)
19. Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., Lerchner, A.: beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In: International Conference on Learning Representations. (2017)
20. Theis, L., Shi, W., Cunningham, A., Huszár, F.: Lossy Image Compression with Compressive Autoencoders. In: International Conference on Learning Representations. (2017)
21. Moreira, L.: Digital video introduction. https://github.com/leandromoreira/digital_video_introduction/blob/master/README.md#frame-types (2017) Accessed: 2020-03-02.
22. Sullivan, G.J., Ohm, J.R., Han, W.J., Wiegand, T.: Overview of the High Efficiency Video Coding (HEVC) Standard. IEEE Trans. Circuits Syst. Video Technol. **22** (2012) 1649–1668
23. Ballé, J., Laparra, V., Simoncelli, E.P.: End-to-End Optimized Image Compression. In: International Conference on Learning Representations. (2017)
24. Ballé, J., Minnen, D., Singh, S., Hwang, S.J., Johnston, N.: Variational Image Compression with a Scale Hyperprior. In: International Conference on Learning Representations. (2018)
25. Mentzer, F., Agustsson, E., Tschannen, M., Timofte, R., Van Gool, L.: Conditional Probability Models for Deep Image Compression. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018)
26. Rippel, O., Bourdev, L.: Real-time adaptive image compression. In: Proceedings of the International Conference on Machine Learning. (2017)
27. Jaderberg, M., Simonyan, K., Zisserman, A., Kavukcuoglu, K.: Spatial Transformer Networks. In: Advances in Neural Information Processing Systems. (2015)
28. Gregor, K., Danihelka, I., Graves, A., Rezende, D.J., Wierstra, D.: DRAW: A Recurrent Neural Network For Image Generation. In: Proceedings of the International Conference on Machine Learning. (2015)
29. Gregor, K., Besse, F., Rezende, D.J., Danihelka, I., Wierstra, D.: Towards Conceptual Compression. In: Advances in Neural Information Processing Systems. (2016)
30. Ronneberger, O., Fischer, P., Brox, T.: U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Medical Image Computing and Computer-Assisted Intervention. (2015)
31. Xue, T., Chen, B., Wu, J., Wei, D., Freeman, W.T.: Video Enhancement with Task-Oriented Flow. International Journal of Computer Vision **127** (2019)
32. Hu, Y., Yang, W., Ma, Z., Liu, J.: Learning end-to-end lossy image compression: A benchmark. arXiv:2002.03711 (2020)
33. Ballé, J., Laparra, V., Simoncelli, E.P.: Density Modeling of Images using a Generalized Normalization Transformation. In: International Conference on Learning Representations. (2016)
34. Liu, H., Chen, T., Guo, P., Shen, Q., Cao, X., Wang, Y., Ma, Z.: Non-local Attention Optimized Deep Image Compression. arXiv:1904.09757 (2019)
35. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press (2009)

36. Alemi, A.A., Poole, B., Fischer, I., Dillon, J.V., Saurous, R.A., Murphy, K.: Fixing a Broken ELBO. In: Proceedings of the International Conference on Machine Learning. (2018)
37. Webb, S., Goliński, A., Zinkov, R., N, S., Rainforth, T., Teh, Y.W., Wood, F.: Faithful Inversion of Generative Models for Effective Amortized Inference. In: Advances in Neural Information Processing Systems. (2018)
38. Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., Suleyman, M., Zisserman, A.: The Kinetics Human Action Video Dataset. arxiv:1705.06950 (2017)
39. Xiph.org: Xiph.org video test media [derf's collection]. https://media.xiph.org/video/derf/ (2004) Accessed: 2020-02-21.
40. Wiegand, T., Sullivan, G.J., Bjontegaard, G., Luthra, A.: Overview of the H.264/AVC video coding standard. IEEE Transactions on Circuits and Systems for Video Technology **13** (2003) 560–576
41. Tomar, S.: Converting video formats with ffmpeg. Linux Journal **2006** (2006) 10
42. HM developers: High Efficiency Video Coding (HEVC). https://hevc.hhi.fraunhofer.de/ (2012) Accessed: 2020-02-21.
43. Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T.: FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2017)
44. Gao, C., Gu, D., Zhang, F., Yu, Y.: ReCoNet: Real-Time Coherent Video Style Transfer Network. In: Proceedings of the Asian Conference on Computer Vision. (2018)
45. Lai, W.S., Huang, J.B., Wang, O., Shechtman, E., Yumer, E., Yang, M.H.: Learning Blind Video Temporal Consistency. In: Proceedings of the European Conference on Computer Vision. (2018)
46. Zhao, H., Gallo, O., Frosio, I., Kautz, J.: Loss Functions for Image Restoration With Neural Networks. In: IEEE Transactions on Computational Imaging. (2017)
47. Bradski, G.: The OpenCV Library. Dr. Dobb's Journal of Software Tools (2000)
48. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: Advances in Neural Information Processing Systems. (2019)
49. Kingma, D., Ba, J.: Adam: A Method for Stochastic Optimization. In: International Conference on Learning Representations. (2015)
50. Wang, Z., Bovik, A.C.: Mean Squared Error: Love it or leave it? A new look at Signal Fidelity Measures. In: IEEE Signal Processing Magazine. Volume 26. (2009) 98–117
51. Minnen, D., Singh, S.: Channel-wise autoregressive entropy models for learned image compression. In: IEEE International Conference on Image Processing (ICIP). (2020)