KAVA: KEYFRAME-ANCHORED VIDEO ADAPTERS FOR EXTREME VIDEO COMPRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Video codecs, classical and learned, transmit whatever their decoder cannot predict from the frames it has already decoded, and at low rates they drop fine details. A pretrained video-diffusion model has learned much of how natural video looks and moves, and if the same copy sits at both ends of the channel, it can predict much of what a codec would otherwise transmit. We present KAVA, a codec that specializes one such frozen model, Wan2.1, to each target video. A standard codec sends sparse, heavily compressed keyframes, and the model generates every frame from them, the keyframe positions included. Because the model alone would produce plausible motion rather than the target's, we fit a rank-one adapter to each video on the layers where tokens enter and leave the transformer and on its text and time conditioning, leaving every transformer block frozen. The bitstream is these keyframes and this adapter alone. On full-length UVG sequences, KAVA delivers state-of-the-art perceptual performance from 0.0015 to 0.0062 bits per pixel against traditional, learned and prior generative codecs. We see specializing such a shared model to each video, so that the bitstream carries only what the model cannot predict, as a key step toward extreme video compression.

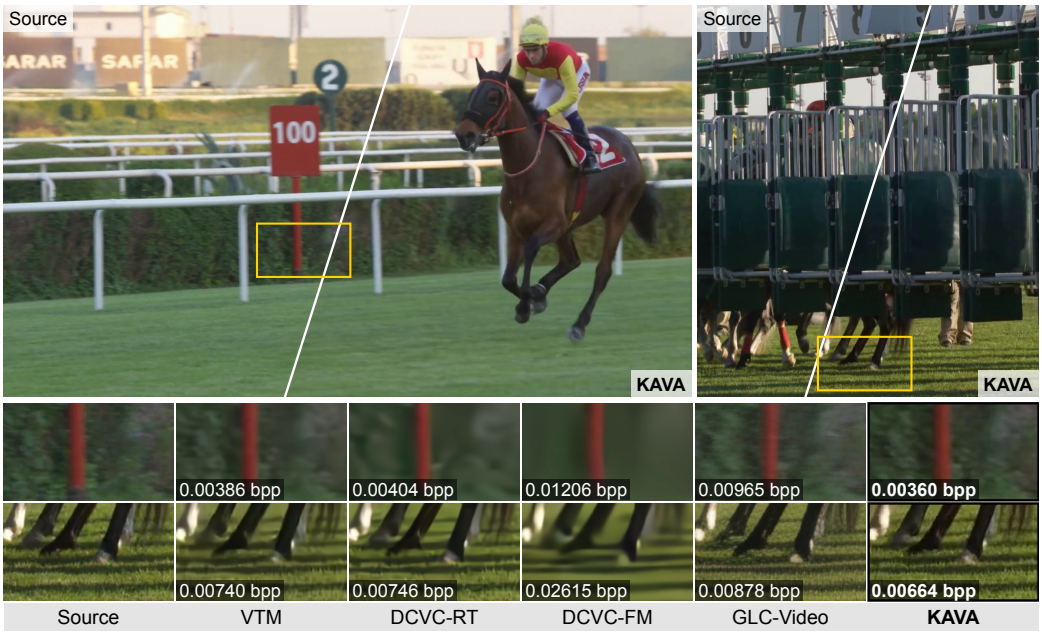

Figure 1: **Qualitative comparison on UVG at** 1920×1088**.** VTM, DCVC-RT and DCVC-FM smooth the hedge leaves and the grass blades into flat patches, and GLC-Video keeps their texture only at higher rates. KAVA's reconstructions keep the leaf, hoof and grass detail of the source at the lowest rate in each row. Rates and settings: Appendix H.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

Classical and learned video codecs predict each frame from frames already decoded and pay for every residual they send, so at low rates they drop the residuals that carry fine details (Figure 1). Much of what such a codec cannot afford to send could already be at the receiver. A pretrained video-diffusion model predicts plausible frames of natural video, and the same copy can sit at both ends of the channel before any bitstream arrives. Learned codecs also hold a trained model at both ends, but it predicts from decoded frames and entropy-codes what is left, so the content still crosses the channel as a coded latent for every frame. Codecs that store a video in the weights of a network fitted to it share no pretrained model, or only one meta-learned on a narrow corpus, so the bitstream pays for everything the network learns. Recent methods, ours included, keep a pretrained generative model frozen at both ends. They differ in what they transmit about the target (Figure 2).

We specialize that frozen model to one target video. Given two boundary frames, the model produces plausible motion between them, but not necessarily the target's. We therefore send two things. Sparse keyframes, heavily compressed by a standard inter-frame codec, anchor the content. A low-rank adapter (LoRA) of rank one adjusts the model to that video. Every frame is generated, so the output at a keyframe position can be closer to the source than the decoded keyframe (Figure 7).

We make three contributions. (i) We introduce KAVA (Keyframe-Anchored Video Adapter), a video codec that specializes a frozen video-generation foundation model, Wan2.1, to each target video. Sparse keyframes, heavily compressed by a standard codec, anchor the content, and a rank-one adapter, fitted once per video and keyframe quality and shared by all its clips, steers the model's interpolation between them toward the target. The bitstream consists of the keyframes and the adapter alone (Section 3.1). (ii) We design the adapter to act only where tokens enter and leave the transformer (the patchify layer and output head, which we call the interface layers) and on its text and time conditioning, leaving all forty transformer blocks frozen. This keeps it at $81{,}360$ parameters, twenty to fifty times fewer than a rank-one LoRA on the attention projections, and the decoder derives each clip's prompt from its index at no rate cost (Sections 3.2 and 3.3). (iii) On full-length sequences of the Ultra Video Group (UVG) set, which are long enough to spread the adapter's fixed cost, KAVA delivers state-of-the-art perceptual performance from 0.0015 to 0.0062 bits per pixel against traditional, learned and prior generative codecs, and saves 23 to 92 percent of the bits over traditional and learned codecs at equal perceptual quality (Section 4.1).

2 RELATED WORK

Learned and traditional codecs. HEVC (Sullivan et al., 2012) and VVC (Bross et al., 2021) transmit a motion-compensated prediction and the residual that corrects it. DVC (Lu et al., 2019) learned that pipeline end to end, DCVC (Li et al., 2021) and its successors DCVC-FM (Li et al., 2024b) and DCVC-RT (Jia et al., 2025) code each frame conditioned on a propagated feature context, and GLC-Video (Qi et al., 2025) codes in a generative latent space. Instance-adaptive compression (van Rozendaal et al., 2021) fine-tunes a pretrained image codec's decoder on one video and sends the parameter update with a latent for every frame.

Storing a video in network weights. NeRV (Chen et al., 2021) fits a network that maps a frame index to the frame, and its compressed weights are the bitstream. HNeRV (Chen et al., 2023a) and HiNeRV (Kwan et al., 2023) refine the architecture, and NVRC (Kwan et al., 2024) optimizes the representation end to end for rate and distortion. COIN (Dupont et al., 2021) stores an image in a small MLP, COIN++ (Dupont et al., 2022) sends modulations of a meta-learned base, and C3 (Kim et al., 2024) fits a latent grid and a small decoder.

Generative priors as decoders. CDC (Yang & Mandt, 2023) decodes transform-coded latents with a conditional diffusion model, PerCo (Careil et al., 2024) conditions a text-to-image model on a quantized latent and a caption, and OneDC (Xue et al., 2025) decodes in one step. For video, GNVC-VD (Mao et al., 2026) refines decoded latents with a video diffusion transformer, and DiffVC (Ma & Chen, 2025) and DeNC (Zhou et al., 2025) restore every coded frame with a diffusion model. Other codecs send only some frames: Li et al. (2024a) predict the next frames from a few neurally coded ones, and Relic et al. (2025) interpolate between coded keyframes with a video diffusion model guided by sparse optical flow.

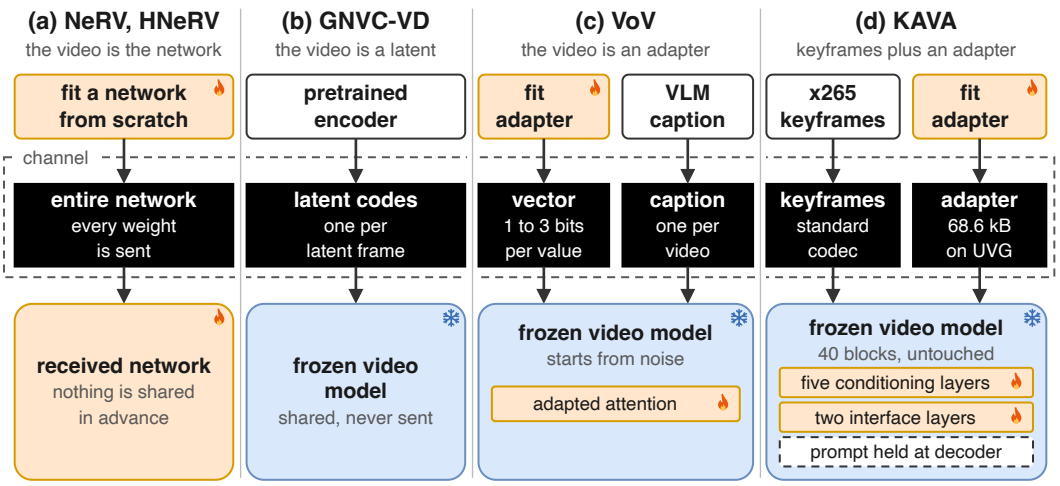

Figure 2: Four ways to code a target video x into a reconstruction $\hat{x}$. Blue (snowflake): shared by encoder and decoder, never transmitted. Orange (flame): fitted to the one video. Black, inside the dashed channel band: what each method sends. (a) A network trained from scratch is transmitted whole (Chen et al., 2021; 2023a). (b) A frozen video model supplies the prior, but the target crosses the channel as coded latents (Mao et al., 2026). (c) A frozen model is adapted by a small transmitted vector and generates from noise and a caption (Guo et al., 2026). (d) We send heavily compressed x265 (MulticoreWare, 2021) keyframes and a rank-one adapter on the interface and conditioning layers.

Weights of a frozen pretrained model as the bitstream. DiV-INR (Çetin et al., 2026) transmits a HiNeRV-style network with NOLA adapters (Koohpayegani et al., 2024) on a frozen Wan2.1, and the adaptation refines that network's coarse reconstruction. VoV (Guo et al., 2026) adapts a frozen video model with LoRA (Hu et al., 2022), generating all attention-projection updates from one vector, and generates from noise and a caption. Unlike both, we fit no per-video network and adapt no attention layer: a standard codec sends keyframes, each clip is conditioned on two of them, and a small adapter on the model's interface and conditioning layers specializes the generator to the video.

3 Method

3.1 Problem Formulation

Let the target be a video $x = (x_0, \ldots, x_{T-1})$ of T frames with $x_i \in \mathbb{R}^{H \times W \times 3}$ at the coding resolution $H \times W$ (Section 4). The receiver holds a frozen generative model Φ that, given a pair of boundary frames, predicts the frames between them; in our instantiation Φ is the first-and-last-frame variant of the Wan2.1 video-diffusion model (Wan Team, 2025). Encoder and decoder both hold it before any bitstream arrives, so in information-theoretic terms it is side information (Gray, 1973; Cover & Thomas, 2006), and our task is to code x at low rate by exploiting it.

The encoder transmits two quantities (Figure 3). The first is a sparse set of keyframes $k_0, \ldots, k_n$ drawn from x (Section 3.3) and heavily compressed as a standard inter-frame bitstream $\mathcal{B}_{\mathrm{kf}}$. The second is an adapter Δ that specializes Φ to x. The decoder recovers the keyframes $\tilde{k}_j = \mathrm{Dec}(\mathcal{B}_{\mathrm{kf}})_j$ with the keyframe decoder Dec, applies Δ to Φ, and generates each clip bounded by two consecutive decoded keyframes,

$$\Phi_\Delta(\tilde{k}_j, \tilde{k}_{j+1}; c_j) = \big(\hat{k}_j,\ \hat{x}_{\tau_j+1},\ \ldots,\ \hat{x}_{\tau_{j+1}-1},\ \hat{k}_{j+1}\big), \tag{1}$$

where τ_j is the source index of keyframe k_j and c_j the clip's text and image conditioning, defined in Eq. (4). The decoded keyframes are only the conditioning input: every frame of the output, the boundary frames $\hat{k}_j$ included, is generated, so the output does not have to keep the keyframes' coding artifacts. Concatenating the clips gives $\hat{x}$, keeping the later clip's frame at each shared boundary. The scheme is an offline, bidirectional codec with independent clips.

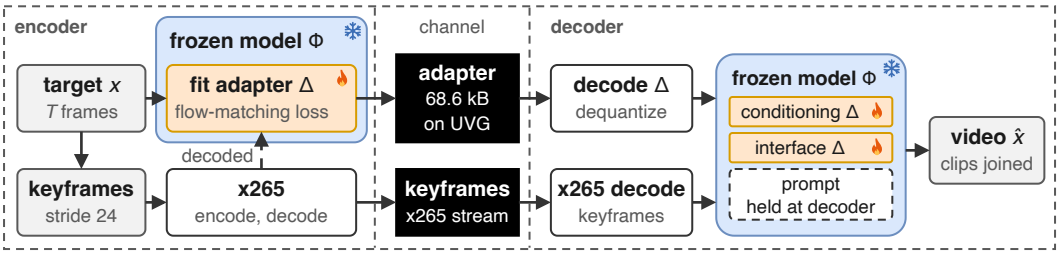

Figure 3: Encoder, channel and decoder. Only the two black bars cross the channel: the adapter Δ (orange), fitted inside the frozen Wan2.1 model Φ (blue), and the x265 keyframe stream. The decoder applies Δ to the same Φ and generates the clips.

The rate in bits per pixel is the total length of the coded adapter $\mathcal{B}_\Delta$ and the keyframe stream $\mathcal{B}_{\text{kf}}$ divided by the number of coded pixels,

$$R = \frac{|\mathcal{B}_\Delta| + |\mathcal{B}_{\text{kf}}|}{N\,H\,W}, \tag{2}$$

where $|\cdot|$ is a length in bits and N is the number of coded frames (Section 3.3). The adapter term is nearly constant per video, while $|\mathcal{B}_{\text{kf}}|$ follows from the constant rate factor (CRF) of the keyframe codec. The CRF fixes the rate, and fitting the adapter by Eq. (5) to the resulting decoded keyframes reduces distortion. Each CRF gives one operating point, and sweeping it traces the operational rate-distortion curve.

3.2 Adapter Placement

The frozen model comprises a diffusion transformer (DiT) with forty blocks, a variational autoencoder (VAE) that compresses a clip by $4\times8\times8$ into a 16-channel latent, a uMT5 text encoder (Chung et al., 2023), and a CLIP image encoder (Radford et al., 2021). This variant conditions on its two boundary frames by stacking their VAE encoding and a binary mask with the noisy target latent along channels (Eq. (5)). We specialize Φ with one low-rank adapter applied to a small set of layers $\mathcal{S}$,

$$\boldsymbol{\Theta}'_\ell = \boldsymbol{\Theta}_\ell + \mathbf{B}_\ell\mathbf{A}_\ell, \qquad \mathbf{B}_\ell \in \mathbb{R}^{d_\ell\times r},\ \mathbf{A}_\ell \in \mathbb{R}^{r\times m_\ell},\ r=1, \quad \ell\in\mathcal{S}, \tag{3}$$

with $\boldsymbol{\Theta}_\ell \in \mathbb{R}^{d_\ell\times m_\ell}$ frozen and only $\Delta = \{\mathbf{A}_\ell, \mathbf{B}_\ell\}_{\ell\in\mathcal{S}}$ trained; for the convolution, m_ℓ is the flattened kernel (shapes in Table 5). The adapter's size, and so its rate, is linear in r.

The budget constrains $\mathcal{S}$. A rank-one LoRA on the blocks' attention projections, the usual target of LoRA, would be twenty to fifty times larger than our adapter (Table 5). We adapt instead the layers where information enters the model (the patchify layer), leaves it (the output head) or conditions it (the entry and refinement layers of the text and time embeddings, and the time projection), and freeze the blocks (Figure 4).

Interface layers. The patchify layer, a 3D convolution, turns the stacked latent into tokens, and the output head maps tokens back to the latent grid the VAE decodes. These two layers translate a video into the representation Wan2.1 operates on and back to the video's own distribution. The output head's shift and scale come from the time embeddings, so the adapter on the time embeddings changes the head as well. We adapt these two layers because we want one approach for any video, including videos far from what Wan2.1 was trained on, whose translation into its representation must then be adjusted; their terms are part of every adapter and counted in its rate. Section 4.2 measures their contribution.

Conditioning embeddings. Each embedding is two linear layers, an entry and a refinement layer. The text embeddings map the encoded prompt to the context tokens every block reads through cross-attention. They act on each token separately, so the adapter learns one video-specific map applied to every prompt token. The time embeddings encode the timestep, and the time projection turns that code into every block's adaptive layer-norm modulation, so adapting them changes that modulation at each denoising step. Both pathways are shared by all clips, so per-clip variation must come from their inputs.

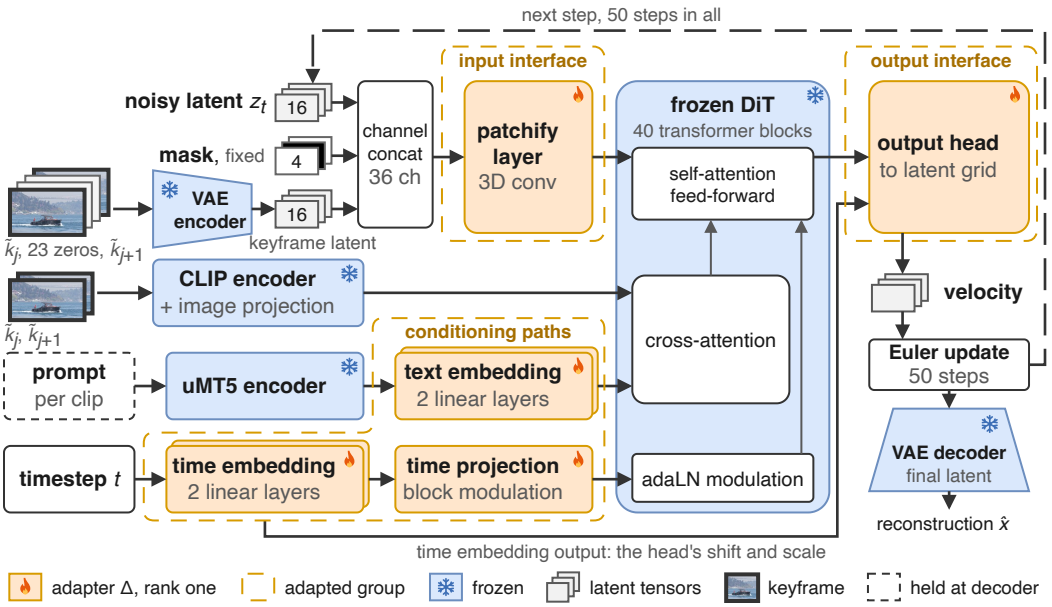

Figure 4: One denoising step of Wan2.1 (the dashed loop repeats it for every Euler step). Orange: the seven rank-one adapted layers, the patchify layer and output head (interface) and the text and time embeddings, two layers each, and the time projection (conditioning). Blue: frozen, including every transformer block.

The adapter totals 81,360 parameters (Table 5). For transmission we apply symmetric per-tensor eight-bit quantization with scale $\max|M|/127$ for each matrix $M \in \Delta$ and code the result and the 14 scales losslessly with the Lempel-Ziv-Markov chain algorithm (LZMA). On UVG every rate charges 68,600 bytes per sequence, the largest coded size of our UVG adapters; on HEVC Class B each operating point is charged the size of its own adapter (Table 12).

3.3 Keyframes and the Prompt

We draw keyframes at a fixed stride, $k_j = x_{\tau_j}$ with $\tau_j = js$, $s = 24$ and $n = \lfloor (T-1)/s \rfloor$. The $N = ns + 1$ frames up to the last keyframe are coded (577 of a 600-frame sequence), and any later frames are not. We code the keyframes as one short video with x265 at its default settings, so most are coded as inter frames (Appendix B).

Each decoded keyframe enters Φ by two routes. The latent route is the channel-stacked input, which the adapted patchify layer tokenizes. The semantic route encodes the two boundary frames with CLIP and prepends their projected embeddings to the text context, so that for clip j

$$c_j = \left[\, \mathrm{MLP}\big([\mathrm{CLIP}(\tilde{k}_j);\, \mathrm{CLIP}(\tilde{k}_{j+1})]\big)\, ;\, \mathrm{TE}_\Delta\big(\mathrm{uMT5}(u_j)\big) \,\right], \tag{4}$$

where $[\,\cdot\,;\,\cdot\,]$ concatenates two token sequences and MLP is the frozen image-embedding projection. TE_Δ denotes the adapted text embeddings. The text u_j joins, in this order, a keyword g_j built from a fixed prefix and the zero-padded clip index (`clip000`, `clip001`, ...), one generic caption p shared by every video and clip, and f_j, a fixed vocabulary $V = (\nu_0, \ldots, \nu_{242})$ of 243 neutral single-token words rotated to start at ν_j: $f_j = (\nu_j, \ldots, \nu_{242}, \nu_0, \ldots, \nu_{j-1})$. The text is padded to 512 tokens (Figure 6 and full text in Appendix A.1). The decoder holds p and V and derives the keyword from the clip index, so the prompt adds no bits. The neutral words fill positions otherwise left as padding, so more positions get their own correction (Eq. (6), Appendix A.1).

3.4 Training and Decoding

For each target we fit Δ with the frozen model's own flow-matching objective (Lipman et al., 2023),

$$\mathcal{L}(\Delta) = \mathbb{E}_{j,t,\epsilon}\, w_t \big\| v_\Delta(\mathrm{cat}(z_t, y_j); t, c_j) - (\epsilon - \bar{z}_j) \big\|_2^2, \qquad z_t = (1-\sigma_t)\,\bar{z}_j + \sigma_t\,\epsilon, \tag{5}$$

where $\bar{z}_j$ is the clean latent of clip j, z_t the noisy latent at step t, ϵ standard Gaussian noise, σ_t the noise level of step t, c_j the conditioning of Eq. (4), and v_Δ the velocity predicted by Φ_Δ. Here cat stacks its arguments along channels, so the model's 36-channel input holds z_t and $y_j = \mathrm{cat}(\mathbf{1}_{\mathrm{kf}}, \mathcal{E}(\tilde{k}_j, 0, \ldots, 0, \tilde{k}_{j+1}))$: a 4-channel mask $\mathbf{1}_{\mathrm{kf}}$ of the boundary frames and the VAE encoding $\mathcal{E}$ of a clip holding the decoded keyframes at its ends and zeros between. The step t is drawn uniformly from the 50 steps of the decoding schedule, and w_t is a fixed per-step weight (both in Appendix B). We train with Lion (Chen et al., 2023b) (hyperparameters in Appendix B). The conditioning is the decoded keyframe at the CRF being trained, as at decode. Each run trains for up to 2,000 epochs over the sequence's clips, and the encoder, which holds the target, transmits the checkpoint with the lowest LPIPS on it. Every result we report, DISTS and PSNR included and ablations included, is measured on that checkpoint.

Because the loss compares the whole clip with the clean target while the conditioning is degraded, the generated boundary frames $\hat{k}_j$ can come closer to the target than the decoded keyframes $\tilde{k}_j$ (examples in Figure 7).

The decoder decodes and dequantizes the adapter and runs it unmerged, at negligible cost (Appendix G). For each clip of Eq. (1) we integrate the flow-matching ODE with 50 first-order Euler steps on that schedule and without classifier-free guidance, from noise seeded by the clip index; one VAE decode follows.

4 Experiments

Sequences and measures. Our primary test set is the seven UVG sequences (Mercat et al., 2020) at 832×480 and 1920×1088, and our secondary one the five of HEVC Class B (Bossen, 2013) at 1920×1088 (Table 6). Every codec we ran encodes and is scored on the frames our scheme codes (Section 3.3), with its bits divided by their number. We report LPIPS (Zhang et al., 2018) (learned perceptual image patch similarity) with VGG features, as GNVC-VD and VoV report it, DISTS (Ding et al., 2022) (deep image structure and texture similarity) and PSNR. The perceptual measures are primary, since distortion and perceptual quality trade off (Blau & Michaeli, 2018); on PSNR (Table 17) the conventional and DCVC codecs score higher than we do. Every run of ours, the adapter-free control (Section 4.1) and the ablations included, uses the prompt of Eq. (4).

Baselines. We ran the HEVC (Sullivan et al., 2012) and VVC reference encoders HM-18.0 and VTM-24.0, x265, an open-source HEVC encoder (MulticoreWare, 2021), DCVC-RT (Jia et al., 2025), DCVC-FM (Li et al., 2024b) and GLC-Video (Qi et al., 2025) from released implementations; GLC-Video's rate is its code's entropy estimate (Appendix B). For GNVC-VD (Mao et al., 2026) and VoV (Guo et al., 2026) we read published figures (Appendix B). VoV specializes a text-to-video model and uses no keyframe; it evaluates one 81-frame clip per sequence at 832×480, and we report its base configuration and VoV+Scaling, its encoding-time candidate search. GNVC-VD evaluates the first 96 frames at native resolution.

A structural asymmetry. Our adapter is a fixed cost per sequence, so it does not amortize over the short prefixes the published methods evaluate: over VoV's frames it alone would cost more than VoV's highest published rate. We therefore compare over each sequence's coded frames, the setting the scheme is built for, and score us on the two methods' frames in Appendix F.

4.1 Main Results

Our operating points sweep the keyframe CRF, five values at 832×480 and three at 1920×1088, with one adapter per sequence and CRF (Appendix C).

Table 1 compares us with each competitor at a matched rate, and Figure 5 compares curves. At 832×480 we have the lowest LPIPS of all methods and the lowest DISTS. On UVG, every operating point of another codec inside our rate range, VoV, VoV+Scaling and GNVC-VD included, lies above our curves on both measures.

At 1920×1088, DCVC-RT, DCVC-FM, HM and x265 spend 1.2 times our rate and are worse on both measures; GNVC-VD spends 1.5 times, is worse on DISTS and ties us on LPIPS within

Table 1: Comparison on UVG. KAVA: our highest-rate point below the lowest published rate of VoV (832×480, CRF 35) and of GNVC-VD (1920×1088, CRF 33). Each codec we ran is at its cheapest setting at or above our rate; for GLC-Video at 1920×1088 that is its lowest setting. In parentheses: rate as a multiple of ours. ‡: read from published figures, over that method's own frames, to ± 0.005 LPIPS and ± 0.003 DISTS. Bold: best.

832×480				1920×1088			
Method	bpp	LPIPS ↓	DISTS ↓	Method	bpp	LPIPS ↓	DISTS ↓
KAVA	0.00507	**0.2682**	**0.0928**	**KAVA**	0.00235	0.2887	**0.0750**
GLC-Video	0.00703 (1.4×)	0.2816	0.1047	GLC-Video	0.00625 (2.7×)	**0.2819**	0.0854
VTM	0.00575 (1.1×)	0.3362	0.1746	VTM	0.00548 (2.3×)	0.2886	0.1263
HM	0.00599 (1.2×)	0.3553	0.1713	HM	0.00278 (1.2×)	0.3593	0.1371
x265	0.00520 (1.0×)	0.3900	0.1859	x265	0.00284 (1.2×)	0.3807	0.1378
DCVC-RT	0.00668 (1.3×)	0.3093	0.1772	DCVC-RT	0.00277 (1.2×)	0.3444	0.1830
DCVC-FM	0.00589 (1.2×)	0.3213	0.1870	DCVC-FM	0.00274 (1.2×)	0.3544	0.1946
VoV+Scaling‡	0.0061 (1.2×)	0.275	0.096	GNVC-VD‡	0.0035 (1.5×)	0.291	0.115
VoV‡	0.0059 (1.2×)	0.428	0.131				

Table 2: BD-rate (%) against each codec we ran, computed per sequence and averaged over sequences; negative means we spend fewer bits at equal quality. *: on some sequences all our operating points are better than all of the competitor's, so no BD-rate exists there and the mean covers the others (Appendix C).

	UVG 832×480		UVG 1920×1088		HEVC-B 1920×1088	
Competitor	LPIPS	DISTS	LPIPS	DISTS	LPIPS	DISTS
HM	−36.4	−60.3	−47.8	−71.8	−62.0	−80.6
VTM	−23.3	−57.4	−40.3	−87.9*	−58.3	−82.3*
x265	−56.1	−71.1	−61.4	−70.9	−64.6	−77.0
DCVC-RT	−47.2	−80.8	−60.1	−92.3*	−57.3	−86.9
DCVC-FM	−44.5	−79.9	−61.3	−91.9*	−61.9	−88.0

reading error. On the mean curves of Figure 5, the other codecs we ran reach our LPIPS only at 2.3 to 2.9 times our rate, and GLC-Video has no setting below 2.7 times; at 2.3 and 2.7 times, VTM and GLC-Video still have 68 and 14 percent higher DISTS than ours.

On the published methods' own frames, with the whole adapter charged to them (Table 18), our DISTS is 28 to 31 percent lower than GNVC-VD's at equal rate but our LPIPS is 0.011 to 0.024 higher, and on VoV's 81 frames none of our points dominates either VoV configuration.

Table 2 gives Bjøntegaard delta rates (BD-rates) (Bjøntegaard, 2001) against every codec we ran, and Table 13 (Appendix C) adds the published methods on curves averaged over sequences. On UVG at 1920×1088 we spend 40 to 61 percent fewer bits than HM, VTM, x265 and the two DCVC codecs at equal LPIPS and 71 to 92 percent fewer at equal DISTS.

HEVC Class B. On HEVC Class B we still spend fewer bits than every codec of Table 2 on both measures. However, GLC-Video, at rates above ours, is better than all our points on both measures, and GNVC-VD reaches lower LPIPS than any of our points and lower DISTS above about 0.005 bits per pixel. The adapter is a fixed cost each sequence pays once, so its share of the rate falls as the sequence grows: on Kimono and ParkScene, 217 coded frames against 577 for most UVG sequences, it costs about 2.7 times more per pixel. That amortization penalty is only a tenth of our rate gap to UVG (0.00027 of 0.00267 bits per pixel at CRF 28); the larger effect is keyframes that cost 1.45 to 1.73 times UVG's bytes per frame on this content (Appendix C).

Removing the adapter. Decoding the same keyframe streams with the adapter removed is worse at all eleven operating points, by 0.126 LPIPS on average (Appendix C). Although the control sends no adapter, at 1920×1088 even its CRF 28 stream, at more than twice the rate of our CRF 38 point, is worse on both measures (Table 9).

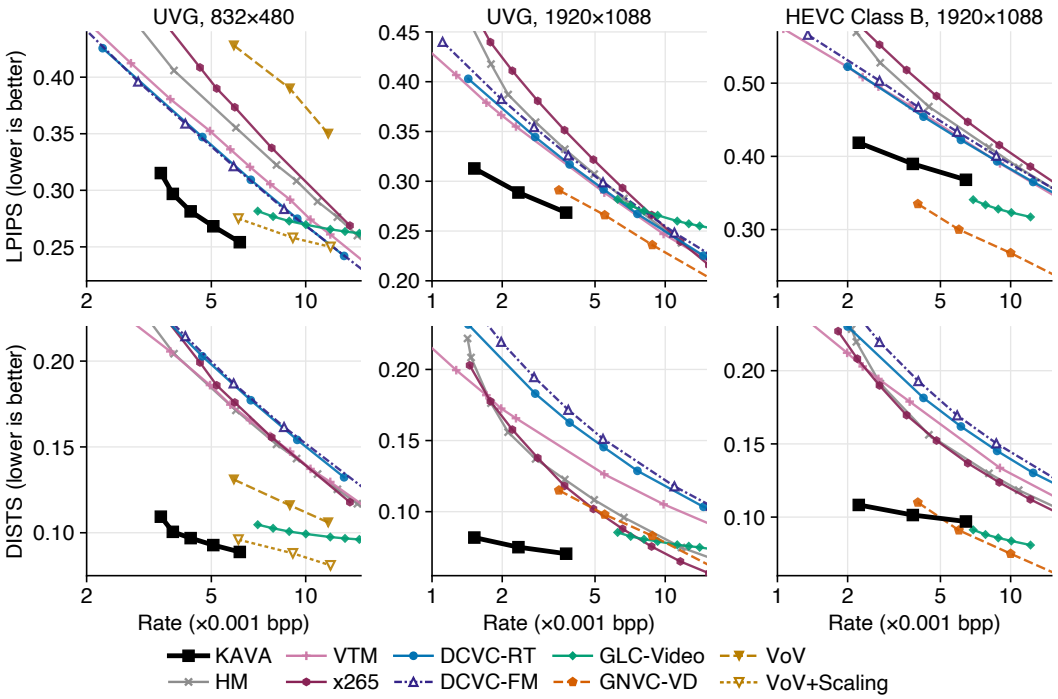

Figure 5: Rate-distortion on UVG at 832×480 and 1920×1088 (seven-sequence means) and on HEVC Class B (five-sequence means). Solid and dash-dotted (DCVC-FM) curves are our measurements of released implementations; dashed or dotted curves are read from published figures (their evaluation protocols in Section 4). Axes are cropped to our operating range; Figure 8 shows every point.

Table 3: Temporal consistency on UVG at 1920×1088, over the coded frames of each sequence; KAVA at CRF 33, each codec at its cheapest setting at or above our rate; VTM not run. Measures: Appendix E.

Method	Rate (bpp)	Warp error ↓	CLIP-F ↑
KAVA	0.00235	**13.35**	**0.9941**
DCVC-RT	0.00277	15.80	0.9928
DCVC-FM	0.00274	16.17	0.9927
HM	0.00278	13.63	0.9930
x265	0.00284	13.77	0.9914
GLC-Video	0.00625	32.86	0.9918

Temporal consistency. Our mean warp error and CLIP-F are better than those of every codec in Table 3. The metrics do expose a defect of the scheme: frame pairs that cross a clip boundary carry about four times the warp error of the others (49.13 against 11.87; Appendix E).

Reconstructions. Appendix H shows texture that HM and VTM blur and KAVA keeps (Figure 12), and errors: on Cactus's calendar the saints' names become text-like marks (Figure 11), and at 832×480 DCVC-RT and GLC-Video render most regions closer to the source (Figure 14).

4.2 Adapter Ablations

All ablations use ShakeNDry at 832×480 and CRF 38, with one run per row. The LoRA rows continue from the CRF-38 adapter (Appendix A.2).

Table 4: Ablations of the five conditioning layers, interface layers excluded. Left: LoRA on the named layers only. Right: smaller adapter types.

Adapted	Params	LPIPS	DISTS	PSNR
Time projection	35,840	0.3124	0.1144	25.48
+ timestep entry	41,216	0.3076	0.1106	25.70
+ text entry	50,432	0.3023	0.1113	25.85
+ refinements (all five)	70,912	**0.2951**	**0.1089**	**26.06**
Timestep pathway	51,456	0.3066	0.1128	25.76
Text pathway	19,456	0.3344	0.1239	24.60
Refinements only	20,480	0.3358	0.1211	24.65

Adapter	Params	LPIPS	DISTS	PSNR
LoRA, r=1	70,912	**0.2951**	**0.1089**	**26.06**
VeRA, q=256	52,480	0.3087	0.1117	25.65
VeRA, q=64	51,520	0.3157	0.1123	25.44
LoRA-XS, q=32	5,120	0.4482	0.1775	20.80
LoRA-XS, q=8	320	0.4501	0.1884	20.68

Which conditioning layers matter. Each row of Table 4 (left) adapts only the conditioning layers it names. No row beats the all-five row on any measure. The time projection matters most: adapted alone it comes within 5.9 percent of the all-five row's LPIPS with 51 percent of its parameters, and the two subsets that leave it frozen are the worst, 13 to 14 percent behind on LPIPS.

Whether a smaller adapter would suffice. VeRA (Kopiczko et al., 2024) sends only scaling vectors against shared random projections of rank q, and LoRA-XS (Bałazy et al., 2025) a $q \times q$ matrix in a basis derived from the frozen weights. Both replace the LoRA of the five conditioning layers and start from scratch (Table 4, right). VeRA at q=256 uses 26 percent fewer parameters, but its LPIPS is 2.8 percent worse than our adapter's even before that adapter's extra training (0.3004, Table 15). The best LPIPS of LoRA-XS is 0.4482, close to the unadapted model's 0.4490 (Table 7).

Interface layers. On ShakeNDry their contribution is small: also training the interface layers lowers LPIPS by 0.0023 and DISTS by 0.0024. Trained alone, they leave LPIPS and PSNR essentially unchanged and lower DISTS (Table 7).

5 Discussion and Conclusion

With heavily compressed keyframes from a standard codec and one rank-one adapter of $81{,}360$ parameters, the frozen Wan2.1 model spends 40 to 61 percent fewer bits at equal LPIPS and 71 to 92 percent fewer at equal DISTS than HM, VTM, x265, DCVC-RT and DCVC-FM on UVG at 1920×1088, and removing the adapter is worse at all eleven operating points, by 0.126 LPIPS on average. The gain is perceptual: on PSNR these codecs score 3.4 to 7.5 dB higher at the 1920×1088 rates of Table 1. On HEVC Class B, where keyframes cost 1.45 to 1.73 times UVG's bytes per frame, GLC-Video, at rates above ours, is better on both measures and GNVC-VD on LPIPS.

Limitations. The adapter is a fixed cost, 48 to 79 percent of our rate at 832×480, and penalizes short sequences (Appendix F). Each clip is conditioned only on its two boundary keyframes and an adapter shared by all clips, so content that enters between them, such as an object, a disocclusion or a scene cut, can only be rendered plausibly, and in our examples it differs from the source (bottom row of Figure 13). Each clip is generated separately, and consecutive clips share only their common keyframe and the adapter, so motion can change abruptly where one clip ends and the next begins: frame pairs across a clip boundary carry about four times the warp error of the others (Figure 10). Encoding takes a median of 3.9 to 9.9 GPU-minutes per frame and decoding 4.0 to 30.4 GPU-seconds, with a 16.40-billion-parameter transformer at both ends (Appendix G).

Outlook. Encoding costs a fraction of VoV's reported 144 GPU-minutes per frame, and a faster generator conditioned on boundary frames would speed up both ends. The adapter is paid once per video, so its share of the rate falls as the video grows, and the method suits long videos coded offline. A natural extension is to let the model use cues from inside a clip, not only its two boundary keyframes, to address the content between them that limits the method.

Reproducibility Statement

Section 3 and Appendix A specify the adapter (Eq. (3) and Table 5), the keyframe sampling and codec, the prompt construction and the training and decoding settings. Appendix B records the invocation of every baseline we ran, Appendix G the frozen components and the median encoding and decoding cost per frame, and Appendix D PSNR. The frozen model is publicly released. We will release the code on acceptance. Each clip's starting noise comes from a seed fixed by its index (Appendix B), so on fixed hardware and software the decode is deterministic.

Ethics Statement

A generative decoder renders detail that the bitstream does not carry, and at these rates such detail can look recorded while differing from the source. For example, a generative decoder can turn small printed text into marks that resemble text, as in Figure 11. Video coded this way should not be used as evidence of what a scene contained.

Use of Large Language Models

We used generative AI tools to implement parts of the software that tracks our experiments and to modify the training code of KAVA. We did not use them to write the evaluation code: every reported number comes from evaluation code we wrote by hand, and the same code scores every codec in this paper, ours included. We also used generative AI tools to draft and polish the paper text and to search for related work we might have missed. We reviewed all AI-assisted work and take responsibility for the final content of this paper.

References

Klaudia Bałazy, Mohammadreza Banaei, Karl Aberer, and Jacek Tabor. LoRA-XS: Low-rank adaptation with extremely small number of parameters. In *European Conference on Artificial Intelligence (ECAI)*, 2025. doi: 10.3233/FAIA251185. arXiv:2405.17604.

Gisle Bjøntegaard. Calculation of average PSNR differences between RD-curves. Technical Report VCEG-M33, ITU-T SG16 Q.6 Video Coding Experts Group (VCEG), Austin, TX, USA, 2001. 13th Meeting.

Yochai Blau and Tomer Michaeli. The perception-distortion tradeoff. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. arXiv:1711.06077.

Frank Bossen. Common test conditions and software reference configurations. Technical Report JCTVC-L1100, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Geneva, Switzerland, 2013. 12th Meeting.

Benjamin Bross, Ye-Kui Wang, Yan Ye, Shan Liu, Jianle Chen, Gary J. Sullivan, and Jens-Rainer Ohm. Overview of the versatile video coding (VVC) standard and its applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(10):3736–3764, 2021. doi: 10.1109/TCSVT.2021.3101953.

Marlène Careil, Matthew J. Muckley, Jakob Verbeek, and Stéphane Lathuilière. Towards image compression with perfect realism at ultra-low bitrates. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.10325.

Eren Çetin, Lucas Relic, Yuanyi Xue, Markus Gross, Christopher Schroers, and Roberto Azevedo. DiV-INR: Extreme low-bitrate diffusion video compression with INR conditioning. *arXiv preprint arXiv:2604.08329*, 2026.

Hao Chen, Bo He, Hanyu Wang, Yixuan Ren, Ser-Nam Lim, and Abhinav Shrivastava. NeRV: Neural representations for videos. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. arXiv:2110.13903.

Hao Chen, Matthew Gwilliam, Ser-Nam Lim, and Abhinav Shrivastava. HNeRV: A hybrid neural representation for videos. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023a. arXiv:2304.02633.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. Symbolic discovery of optimization algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023b.

Hyung Won Chung, Noah Constant, Xavier Garcia, Adam Roberts, Yi Tay, Sharan Narang, and Orhan Firat. UniMax: Fairer and more effective language sampling for large-scale multilingual pretraining. In *International Conference on Learning Representations (ICLR)*, 2023.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 2nd edition, 2006.

Keyan Ding, Kede Ma, Shiqi Wang, and Eero P. Simoncelli. Image quality assessment: Unifying structure and texture similarity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(5):2567–2581, 2022. doi: 10.1109/tpami.2020.3045810. arXiv:2004.07728.

Emilien Dupont, Adam Golinski, Milad Alizadeh, Yee Whye Teh, and Arnaud Doucet. COIN: Compression with implicit neural representations. In *Neural Compression Workshop at ICLR*, 2021.

Emilien Dupont, Hrushikesh Loya, Milad Alizadeh, Adam Goliński, Yee Whye Teh, and Arnaud Doucet. COIN++: Neural compression across modalities. *Transactions on Machine Learning Research (TMLR)*, 2022. arXiv:2201.12904.

Robert M. Gray. A new class of lower bounds to information rates of stationary sources via conditional rate-distortion functions. *IEEE Transactions on Information Theory*, 19(4):480–489, 1973.

Zongyu Guo, Jiajun He, Zhaoyang Jia, Xiaoyi Zhang, Jiahao Li, Xiao Li, Bin Li, José Miguel Hernández-Lobato, and Yan Lu. Compression as adaptation: Implicit visual representation with diffusion foundation models. In *International Conference on Machine Learning (ICML)*, 2026. arXiv:2603.07615.

Christian Herglotz, Hannah Och, Anna Meyer, Geetha Ramasubbu, Lena Eichermüller, Matthias Kränzler, Fabian Brand, Kristian Fischer, Dat Thanh Nguyen, Andy Regensky, and André Kaup. The Bjøntegaard bible: Why your way of comparing video codecs may be wrong. *IEEE Transactions on Image Processing*, 33:987–1001, 2024. doi: 10.1109/TIP.2023.3346695.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

Zhaoyang Jia, Bin Li, Jiahao Li, Wenxuan Xie, Linfeng Qi, Houqiang Li, and Yan Lu. Towards practical real-time neural video compression. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. arXiv:2502.20762.

Hyunjik Kim, Matthias Bauer, Lucas Theis, Jonathan Richard Schwarz, and Emilien Dupont. C3: High-performance and low-complexity neural compression from a single image or video. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. arXiv:2312.02753.

Soroush Abbasi Koohpayegani, K L Navaneet, Parsa Nooralinejad, Soheil Kolouri, and Hamed Pirsiavash. NOLA: Compressing LoRA using linear combination of random basis. In *International Conference on Learning Representations (ICLR)*, 2024.

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki Markus Asano. VeRA: Vector-based random matrix adaptation. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.11454.

Ho Man Kwan, Ge Gao, Fan Zhang, Andrew Gower, and David Bull. HiNeRV: Video compression with hierarchical encoding-based neural representation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Ho Man Kwan, Ge Gao, Fan Zhang, Andrew Gower, and David Bull. NVRC: Neural video representation compression. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Bohan Li, Yiming Liu, Xueyan Niu, Bo Bai, Wei Han, Lei Deng, and Deniz Gündüz. Extreme video compression with prediction using pre-trained diffusion models. In *International Conference on Wireless Communications and Signal Processing (WCSP)*, 2024a. doi: 10.1109/WCSP62071.2024.10827256.

Jiahao Li, Bin Li, and Yan Lu. Deep contextual video compression. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Jiahao Li, Bin Li, and Yan Lu. Neural video compression with feature modulation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024b.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2210.02747.

Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. DVC: An end-to-end deep video compression framework. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Wenzhuo Ma and Zhenzhong Chen. Diffusion-based perceptual neural video compression with temporal diffusion information reuse. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 21(12):345:1–345:22, 2025. doi: 10.1145/3761815.

Qi Mao, Hao Cheng, Tinghan Yang, Libiao Jin, and Siwei Ma. Generative neural video compression via video diffusion prior. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. arXiv:2512.05016.

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 (MMSys)*, pp. 297–302. ACM, 2020. doi: 10.1145/3339825.3394937.

MulticoreWare. x265 HEVC encoder. `https://www.x265.org`, 2021. version 3.5.

Linfeng Qi, Zhaoyang Jia, Jiahao Li, Bin Li, Houqiang Li, and Yan Lu. Generative latent coding for ultra-low bitrate image and video compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 2025. doi: 10.1109/tcsvt.2025.3571944. arXiv:2505.16177.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, 2021.

Lucas Relic, André Emmenegger, Roberto Azevedo, Yang Zhang, Markus Gross, and Christopher Schroers. Spatiotemporal diffusion priors for extreme video compression. In *Picture Coding Symposium (PCS)*, pp. 1–5, 2025. doi: 10.1109/PCS65673.2025.11417607.

Jacob Ström, Kenneth Andersson, Rickard Sjöberg, Andrew Segall, Frank Bossen, Gary J. Sullivan, Jens-Rainer Ohm, and Alexis Tourapis. Working practices using objective metrics for evaluation of video coding efficiency experiments. Technical Paper HSTP-VID-WPOM, ITU-T, July 2020. URL `https://www.itu.int/pub/T-TUT-ASC-2020-HSTP1`. Technically aligned with ISO/IEC TR 23002-8:2020.

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. doi: 10.1109/TCSVT.2012.2221191.

Narayanan Sundaram, Thomas Brox, and Kurt Keutzer. Dense point trajectories by GPU-accelerated large displacement optical flow. In *European Conference on Computer Vision (ECCV)*, 2010. doi: 10.1007/978-3-642-15549-9_32.

Zachary Teed and Jia Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *European Conference on Computer Vision (ECCV)*, 2020. arXiv:2003.12039.

Ties van Rozendaal, Iris A. M. Huijben, and Taco S. Cohen. Overfitting for fun and profit: Instance-adaptive data compression. In *International Conference on Learning Representations (ICLR)*, 2021.

Wan Team. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Naifu Xue, Zhaoyang Jia, Jiahao Li, Bin Li, Yuan Zhang, and Yan Lu. One-step diffusion-based image compression with semantic distillation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2505.16687.

Ren Yang, Fabian Mentzer, Luc Van Gool, and Radu Timofte. Learning for video compression with hierarchical quality and recurrent enhancement. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6627–6636, 2020. doi: 10.1109/CVPR42600.2020.00666.

Ruihan Yang and Stephan Mandt. Lossy image compression with conditional diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2209.06950.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. arXiv:1801.03924.

Qihua Zhou, Ruibin Li, Jingcai Guo, Yaodong Huang, Zhenda Xu, Laizhong Cui, and Song Guo. DeNC: Unleash neural codecs in video streaming with diffusion enhancement. In *AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1192–1200, 2025. doi: 10.1609/aaai.v39i1.32107.

A Method Details

A.1 Adapted Layers, Prompt and Sequences

Table 5: The seven adapted layers, all at rank one and LoRA scale 1. For comparison, a rank-one LoRA on the blocks' self-attention projections would have about 1.6 million parameters, and 4.1 million with the cross-attention projections.

Adapted layer	Frozen layer	Parameters
Patchify layer	Conv3d$(36 \to 5120)$, patch $(1, 2, 2)$	5,264
Output head	Linear$(5120 \to 64)$	5,184
Text embedding, entry layer	Linear$(4096 \to 5120)$	9,216
Text embedding, refinement layer	Linear$(5120 \to 5120)$	10,240
Time embedding, entry layer	Linear$(256 \to 5120)$	5,376
Time embedding, refinement layer	Linear$(5120 \to 5120)$	10,240
Time projection	Linear$(5120 \to 30720)$	35,840
Total		81,360

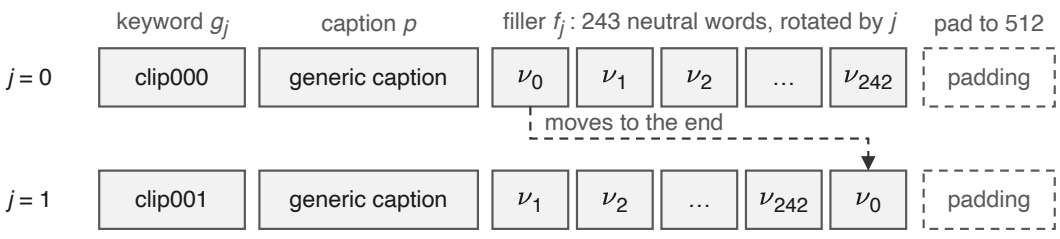

Figure 6: Per-clip prompt: keyword g_j, generic caption p and the neutral words of V rotated by j, padded to 512 tokens.

Prompt text. The keyword g_j is `clip` followed by the clip index in three digits (`clip000`, `clip001`, ...). The generic caption p is "A high-resolution source video clip showing natural continuous motion and realistic visual detail, captured with a standard camera at a smooth frame rate." The vocabulary V, in its order before rotation (ν_0 first), is: structure surface pattern element region segment layer frame section area zone field space volume block mass texture edge contour cluster node unit object item form shape density frequency motion shift flow drift curve angle depth channel band range level phase stage state mode type class group set pair point tone hue shade beam ray flash shadow fabric metal glass stone wood plastic paper rubber wire rotation pan zoom arc spiral wave pulse beat scale ratio proportion degree extent magnitude interval gap margin border center origin vector matrix quality feature aspect attribute property trait factor signal noise contrast grain mist fog patch grid dot mark trace print stamp seal crack split fold valley peak base dust sand salt smoke draft wind spark strand thread fiber particle fragment pellet shell hull case cover lid cap ring coil magnet disc drum roller guide rail track panel plate sheet strip slab column coating film membrane weight height width length size count total sum mean norm rate speed tempo pitch note echo focus lens mirror filter exposure gain balance load force stress strain pressure blot spot hint touch drop index tag label flag code lock gate port hub link bridge tunnel shaft slot tab pin joint bond paste wax resin paint polish line plumb meter probe sensor rotor lever pedal cam bump mesh pile nap limit bound threshold flux score grade rank tier rung cycle loop pass run trial scan sample clone copy cache buffer queue stack pool.

Why neutral words. With $\mathbf{a}_1^\top, \mathbf{b}_1$ and $\mathbf{a}_2^\top, \mathbf{b}_2$ the rank-one factors ($\mathbf{A}_\ell = \mathbf{a}^\top$, $\mathbf{B}_\ell = \mathbf{b}$ in Eq. (3)) of the text embedding's entry and refinement layers, whose frozen layers are TE_1 and TE_2, a position with encoded vector $\mathbf{e}$ receives

$$\mathrm{TE}_\Delta(\mathbf{e}) = \mathrm{TE}_2(\mathbf{h}) + (\mathbf{a}_2^\top \mathbf{h})\,\mathbf{b}_2, \qquad \mathbf{h} = \mathrm{GELU}\big(\mathrm{TE}_1(\mathbf{e}) + (\mathbf{a}_1^\top \mathbf{e})\,\mathbf{b}_1\big). \tag{6}$$

Each distinct $\mathbf{e}$ thus receives its own scalar $\mathbf{a}_1^\top \mathbf{e}$, and these scalars can be set independently when the vectors are linearly independent in the 4096-dimensional input. At padded positions uMT5 outputs $\mathbf{e} = \mathbf{0}$, so the first correction vanishes, every padded position has the same $\mathbf{h}$, and all of them receive the same output. The keyword and the generic caption occupy about 35 of the 512 positions, and the 243 neutral words raise that to about 280. uMT5 computes each vector from the whole sequence with relative position biases. The rotation therefore gives each clip different vectors, and the one video-level mapping turns them into different conditioning.

Table 6: The evaluation sequences, 1920×1080 at source. Clips are 25 frames at stride 24; coded-frame counts in Appendix C.

Set	Sequence	Frames	Clips	Set	Sequence	Frames	Clips
UVG	Beauty	600	24	HEVC-B	BasketballDrive	500	20
	Bosphorus	600	24		BQTerrace	600	24
	HoneyBee	600	24		Cactus	500	20
	Jockey	600	24		Kimono	240	9
	ReadySetGo	600	24		ParkScene	240	9
	ShakeNDry	300	12				
	YachtRide	600	24				

A.2 Adapter Ablations

In every row of Table 4 the adapter terms of the two interface layers keep their values from the CRF-38 adapter. The VeRA and LoRA-XS terms start from scratch at a learning rate of 2×10^{-3}, or 5×10^{-3} for VeRA at q=256; q is VeRA's projection rank and LoRA-XS's matrix size, and neither method applies to the 3D convolution of the patchify layer. LoRA-XS at q=8 never improves on its first validation. In the row names, the entry and refinement layers are the first and second linear layers of the text and time embeddings (Table 5), and a pathway is all layers of one embedding, the timestep one with the time projection.

Table 7: Interface ablation on ShakeNDry, 832×480, CRF 38, one 2,000-epoch run per row. Top: the all-five row of Table 4 against a run from the same CRF-38 adapter that also trains the interface layers. Bottom: an adapter on the interface layers only, against the unadapted model.

Trained layers	Trained parameters	LPIPS	DISTS	PSNR
All five conditioning layers	70,912	0.2951	0.1089	26.06
+ interface layers	81,360	0.2928	0.1065	26.16
None (unadapted)	0	0.4490	0.1886	20.53
Interface only	10,448	0.4475	0.1714	20.57

A.3 Output at Keyframe Positions



Figure 7: The model's output at three keyframe positions (Section 3.4), 832×480, CRF 44: the decoded keyframe it is conditioned on, its output at that position, and the source, with the region's LPIPS. In each, the output is closer to the source than the decoded keyframe, though not faithful in detail: in ReadySetGo the spectator behind the horse is missing.

B Implementation and Baselines

KAVA. The frozen model is the released Wan2.1-FLF2V-14B checkpoint (Wan Team, 2025), whose components Table 19 lists. Keyframes are coded with x265 (version 3.5) through FFmpeg's libx265 wrapper with 4:2:0 chroma and the default preset (medium) and group-of-pictures structure, so the keyframes form one inter-coded stream, at CRF 28, 33 and 38 at 1920×1088 and 32, 35, 38, 41 and 44 at 832×480. The starting noise of clip j uses the seed $42(j+1)$, which encoder and decoder share. Training and decoding use the same 50 noise levels, $\sigma_t = 5\rho_t/(1+4\rho_t)$ with $\rho_t = 1 - t/50$ for steps $t = 0, \ldots, 49$, a linear schedule warped by a shift of 5. The loss weight of

Eq. (5) is the one built into the flow-matching scheduler of the training library; over the 50 steps w_t runs from 0.03 at $\sigma = 1$ to 2.1 at $\sigma \approx 0.49$.

Training. We train with Lion at a learning rate of 2×10^{-4}, momentum parameters $(0.9, 0.99)$ and batch size one, on 25-frame clips at the keyframe stride, from the library's default initialization (Kaiming-uniform $\mathbf{A}_\ell$, zero $\mathbf{B}_\ell$). Each UVG sequence is trained from initialization at CRF 38 and 832×480; its CRF-38 run at 1920×1088 starts from that adapter, the HEVC Class B CRF-38 runs start from Beauty's CRF-38 adapter at 1920×1088, and every other operating point starts from a saved checkpoint of the CRF-38 run of its sequence and resolution.

Codecs we ran. HM-18.0 and VTM-24.0 use the low-delay configuration files distributed with each encoder at both resolutions, on the 8-bit 4:2:0 source with Main-profile files; GNVC-VD runs HM-16.25 and VTM-17.0 with 10-bit 4:4:4 range-extension configurations. x265 runs through FFmpeg at its defaults. DCVC-RT, DCVC-FM and GLC-Video run with their released checkpoints and an intra period of 96 frames, the group size GNVC-VD sets for the two DCVC codecs. GLC-Video writes no bitstream, so its rate is the entropy estimate its implementation reports. The x265 rates, and our own keyframe rates, count the size of the MP4 file, which includes about 3.5 kB of container overhead per stream; the HM, VTM and learned-codec rates count elementary streams only. Every codec is scored with the same code, clip windows and reference video as our reconstructions. The swept settings are in Table 8.

Table 8: Swept settings of the codecs we ran: QP for HM and VTM, CRF for x265, quality index for the learned codecs. [a]: six of the seven sequences.

Codec	UVG 832×480	UVG 1920×1088	HEVC Class B
HM	27, 29, 31, 33 to 38, 40, 43, 46	28, 32, 36, 38, 40, 42, 44, 46, 49, 51	28, 32, 36, 38, 42, 46, 49, 51
VTM	28, 30, 32, 34 to 40, 42, 44, 47, 50	24, 26[a], 28, 32, 36, 42, 43, 44, 46, 50, 54, 58, 63	28, 32, 36, 42, 44, 45, 46, 50, 54, 58, 63
x265	24, 28, 32, 36, 40, 42, 43, 44, 48	32 to 48 in steps of 2, 51	30, 33, 36 to 48 in steps of 2, 51
DCVC-RT and -FM	0, 12, 18, 24, 30, 42, 48, 55, 63	as at 832×480	as at 832×480
GLC-Video	0, 8, 16, 24, 36, 44, 52, 63	as at 832×480	0, 8, 16, 24, 36

VoV. We ran VoV's released code on the seven UVG and five HEVC Class B sequences at 832×480 over 81-frame clips, without candidate search and with the repository's launcher settings, cheaper than the configuration in VoV's appendix. We report VoV's published figures, not this run, which supplies only the VoV* panels of Appendix H. GNVC-VD releases no code, so we also take its published figures.

C Operating Points and Per-Sequence Results

Tables 9, 10 and 11 give our operating points and, below each, the same keyframe streams decoded with the adapter removed; Tables 14, 15 and 16 give the per-sequence values. All values of a point come from its transmitted checkpoint (Section 3.4).

Rate denominators. The denominator is $N\,H\,W$ in Eq. (2). Our rates and those of every codec we ran divide the bits by the frames coded and scored, the counts that follow from the clips of Table 6 as 24 coded frames per clip plus the last keyframe. At full resolution all our rates divide by 1920×1088 pixels per frame, while GNVC-VD reports its rates at the native 1920×1080; against its published figures our rates are lower by 0.74 percent for that reason. The references also differ: our metrics compare against the source cropped to the coding aspect ratio and resized to 1920×1088, while GNVC-VD scores against the native frames.

BD-rate computation. Both tables fit each rate-quality curve as log rate against the quality measure by piecewise cubic Hermite (PCHIP) interpolation and average the rate difference over the over-

Table 9: Operating points on UVG at 1920×1088, seven-sequence means. Adapter share: the per-sequence mean of the 68,600-byte charge over that sequence's total bits. Adapter-removed rows send only the keyframes (mean adapter cost 0.00052 bpp).

Keyframes	Rate (bpp)	LPIPS ↓	DISTS ↓	PSNR ↑	Adapter share
CRF 28	0.00376	0.2685	0.0704	27.07	18%
CRF 33	0.00235	0.2887	0.0750	26.65	28%
CRF 38	0.00152	0.3128	0.0818	25.98	39%
Adapter removed (frozen model, same keyframe streams)					
CRF 28	0.00324	0.3254	0.0904	22.88	–
CRF 33	0.00183	0.3536	0.1052	22.70	–
CRF 38	0.00100	0.3943	0.1295	22.48	–

Table 10: Operating points on UVG at 832×480, seven-sequence means. Adapter share and adapter-removed rates as in Table 9, with a mean adapter cost of 0.00272 bpp.

Keyframes	Rate (bpp)	LPIPS ↓	DISTS ↓	PSNR ↑	Adapter share
CRF 32	0.00616	0.2541	0.0888	25.79	48%
CRF 35	0.00507	0.2682	0.0928	25.29	56%
CRF 38	0.00429	0.2814	0.0969	24.82	65%
CRF 41	0.00377	0.2968	0.1005	24.17	73%
CRF 44	0.00345	0.3152	0.1094	23.66	79%
Adapter removed (frozen model, same keyframe streams)					
CRF 32	0.00344	0.4010	0.1502	19.54	–
CRF 35	0.00235	0.4250	0.1681	19.53	–
CRF 38	0.00157	0.4543	0.1877	19.39	–
CRF 41	0.00105	0.4885	0.2128	19.20	–
CRF 44	0.00073	0.5252	0.2438	18.96	–

lap of the two quality ranges, following the working practices of the video coding standards (Ström et al., 2020) and the PCHIP fit examined by Herglotz et al. (2024). Table 2 (Section 4.1) computes one BD-rate per sequence and reports their mean, as those working practices do for each class of sequences (Ström et al., 2020). Where the quality ranges of a sequence do not overlap, its BD-rate is undefined and the sequence is left out of the mean, as in Herglotz et al. (2024). In Table 2 this happens only where all our operating points are better than all of the competitor's, all on DISTS at 1920×1088 (cells marked *): on UVG, Beauty and HoneyBee against VTM and DCVC-RT, and these two and Jockey against DCVC-FM; on HEVC Class B, BQTerrace against VTM. Table 13 computes one BD-rate per cell from the curves averaged over sequences, the only computation possible for curves read from published figures; Yang et al. (2020) use it for the same reason, for a competitor without per-sequence results. Averaging curves first gives the larger saving in 28 of the 30 cells of Table 2, by up to 31 points (VTM at 832×480 on LPIPS).

Table 11: Operating points on HEVC Class B at 1920×1088, five-sequence means. Adapter share and adapter-removed rates as in Table 9, with each point charged its coded adapter size (Table 12); the mean adapter cost is 0.00084, 0.00085 and 0.00078 bpp at CRF 28, 33 and 38.

Keyframes	Rate (bpp)	LPIPS ↓	DISTS ↓	PSNR ↑	Adapter share
CRF 28	0.00643	0.3680	0.0970	23.59	13%
CRF 33	0.00381	0.3897	0.1014	23.13	22%
CRF 38	0.00223	0.4186	0.1083	22.19	34%
Adapter removed (frozen model, same keyframe streams)					
CRF 28	0.00559	0.4558	0.1280	19.41	–
CRF 33	0.00296	0.4939	0.1480	19.34	–
CRF 38	0.00145	0.5361	0.1731	19.36	–

Table 12: Coded adapter size in bytes (LZMA, Section 3.2) charged to each HEVC Class B operating point.

Sequence	CRF 28	CRF 33	CRF 38
BasketballDrive	71,780	74,476	66,968
BQTerrace	73,020	71,032	66,072
Cactus	71,244	73,124	67,384
Kimono	74,264	73,752	68,748
ParkScene	72,880	72,524	67,176

Table 13: Bjøntegaard delta rate (BD-rate, %; Bjøntegaard, 2001) against each competitor at equal perceptual quality, from curves averaged over sequences (Appendix C); negative means we spend fewer bits. †: overlap under half the quality range of our operating points. n/a: not applicable, because the two quality ranges do not overlap: GNVC-VD on HEVC Class B is better than all our operating points, and all our operating points are better than VoV. –: data not available, because the method reports no results on that set or at that resolution. ‡: read from published figures, over that method's own frames (Section 4).

	UVG 832×480		UVG 1920×1088		HEVC-B 1920×1088	
Competitor	LPIPS	DISTS	LPIPS	DISTS	LPIPS	DISTS
HM	-62.8	-79.3	-61.9	-80.3	-63.5	-80.0
VTM	-54.3	-80.7	-56.9	-91.4	-58.6	-84.2
x265	-64.2	-78.3	-65.9	-74.2	-67.9	-78.0
DCVC-RT	-49.4	-83.1	-58.3	-93.8	-58.7	-87.5
DCVC-FM	-48.9	-83.2	-62.0	-94.7	-62.1	-88.4
GNVC-VD‡	–	–	-33.4	-78.8	n/a	-23.5
VoV+Scaling‡	$-30.9^{\dagger}$	$-29.6^{\dagger}$	–	–	–	–
VoV‡	n/a	$-69.2^{\dagger}$	–	–	–	–

HEVC Class B rates. At CRF 28 our HEVC Class B rate, 0.00643 bits per pixel, exceeds UVG's by 0.00267: the shorter sequences account for 0.00027, the larger adapters for 0.00005, and the keyframes for the rest, since at the same CRF they take 1.45 (CRF 38) to 1.73 (CRF 28) times UVG's bytes per frame, averaged over sequences.

Table 14: Per-sequence values on UVG at 1920×1088. Rates charge the $68{,}600$-byte adapter and divide by the coded frames; their means, taken before rounding, are those of Table 9.

	Rate (bpp)			LPIPS ↓			DISTS ↓		
Sequence	CRF 28	CRF 33	CRF 38	CRF 28	CRF 33	CRF 38	CRF 28	CRF 33	CRF 38
Beauty	0.00193	0.00132	0.00096	0.3472	0.3536	0.3570	0.1149	0.1150	0.1166
Bosphorus	0.00329	0.00186	0.00119	0.2385	0.2619	0.2948	0.0539	0.0597	0.0734
HoneyBee	0.00113	0.00084	0.00071	0.1202	0.1369	0.1539	0.0474	0.0520	0.0557
Jockey	0.00366	0.00242	0.00160	0.3306	0.3528	0.3718	0.0864	0.0952	0.0994
ReadySetGo	0.00682	0.00429	0.00267	0.2657	0.2858	0.3148	0.0533	0.0560	0.0637
ShakeNDry	0.00374	0.00236	0.00164	0.2659	0.2956	0.3394	0.0749	0.0790	0.0900
YachtRide	0.00574	0.00333	0.00186	0.3114	0.3342	0.3583	0.0623	0.0682	0.0738

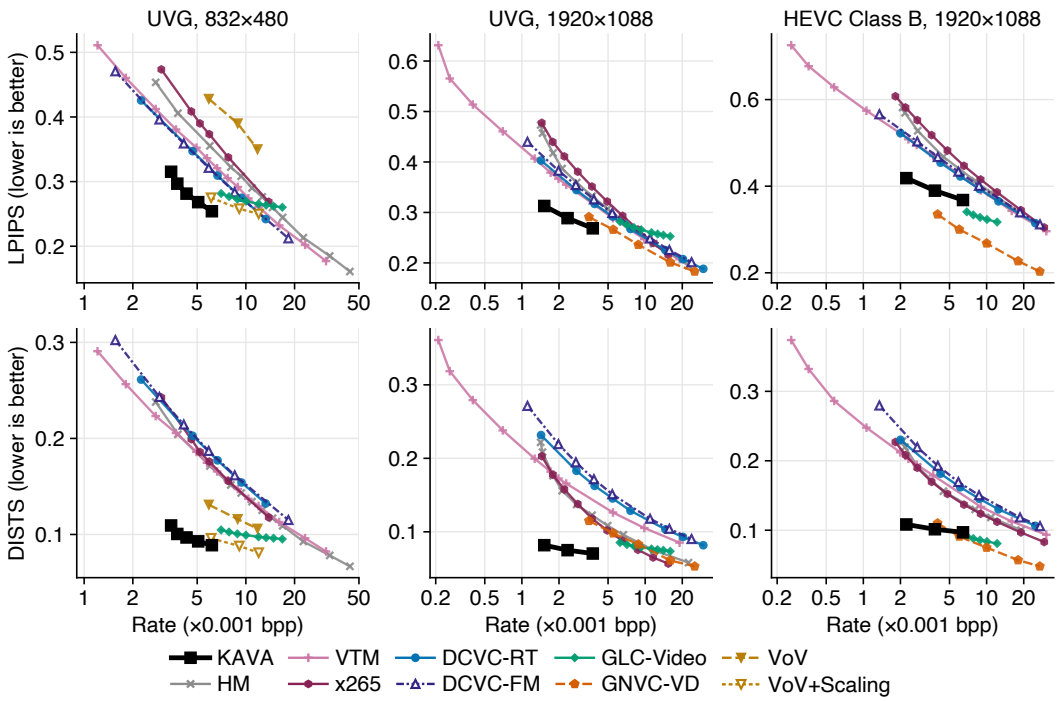

Figure 8: Full-range rate-distortion: every point of the curves of Figure 5, with rate on a log scale; frames and line styles as there.

Table 15: Per-sequence values on UVG at 832×480. Rates as in Table 14; their means, taken before rounding, are those of Table 10.

	CRF 32	CRF 35	CRF 38	CRF 41	CRF 44
Rate (bpp)					
Beauty	0.00416	0.00366	0.00330	0.00305	0.00287
Bosphorus	0.00481	0.00391	0.00333	0.00296	0.00277
HoneyBee	0.00342	0.00318	0.00301	0.00288	0.00278
Jockey	0.00629	0.00522	0.00440	0.00377	0.00337
ReadySetGo	0.00954	0.00740	0.00573	0.00455	0.00387
ShakeNDry	0.00763	0.00670	0.00609	0.00569	0.00543
YachtRide	0.00728	0.00544	0.00422	0.00347	0.00305
LPIPS ↓					
Beauty	0.3049	0.3065	0.3119	0.3172	0.3220
Bosphorus	0.2348	0.2497	0.2643	0.2806	0.3131
HoneyBee	0.0946	0.1037	0.1106	0.1330	0.1306
Jockey	0.3227	0.3345	0.3507	0.3615	0.3809
ReadySetGo	0.2782	0.3004	0.3163	0.3413	0.3805
ShakeNDry	0.2553	0.2810	0.3004	0.3153	0.3384
YachtRide	0.2884	0.3016	0.3154	0.3288	0.3413
DISTS ↓					
Beauty	0.1198	0.1234	0.1229	0.1272	0.1312
Bosphorus	0.0862	0.0901	0.0968	0.0994	0.1150
HoneyBee	0.0460	0.0507	0.0519	0.0586	0.0613
Jockey	0.1060	0.1110	0.1145	0.1157	0.1247
ReadySetGo	0.0857	0.0876	0.0948	0.1002	0.1177
ShakeNDry	0.0981	0.1052	0.1108	0.1134	0.1177
YachtRide	0.0800	0.0818	0.0864	0.0893	0.0980

Table 16: Per-sequence values on HEVC Class B at 1920×1088. Rates charge each point's coded adapter size (Table 12) and divide by the coded frames; their means, taken before rounding, are those of Table 11.

	Rate (bpp)			LPIPS ↓			DISTS ↓		
Sequence	CRF 28	CRF 33	CRF 38	CRF 28	CRF 33	CRF 38	CRF 28	CRF 33	CRF 38
BasketballDrive	0.00553	0.00343	0.00202	0.4114	0.4302	0.4522	0.1197	0.1250	0.1309
BQTerrace	0.00435	0.00238	0.00134	0.3162	0.3345	0.3587	0.0852	0.0876	0.0935
Cactus	0.00490	0.00291	0.00164	0.3240	0.3470	0.3820	0.0635	0.0676	0.0746
Kimono	0.00717	0.00468	0.00303	0.4233	0.4421	0.4717	0.1180	0.1231	0.1335
ParkScene	0.01021	0.00564	0.00314	0.3650	0.3946	0.4282	0.0986	0.1037	0.1090

D FIDELITY

Table 17 and Figure 9 give PSNR. HM, VTM, x265 and the two DCVC codecs lead us by 2.5 to 7.5 dB, the direction the perception-distortion trade-off predicts. Among GLC-Video, VoV and GNVC-VD we are above GLC-Video and both VoV configurations at 832×480, and 2.6 dB below GLC-Video and GNVC-VD at 1920×1088.

Table 17: PSNR (dB) at the operating points of Table 1, rows in the same order. ‡: read from published figures, about ± 0.1 dB.

832×480			1920×1088		
Method	bpp	PSNR	Method	bpp	PSNR
KAVA	0.00507	25.29	KAVA	0.00235	26.65
GLC-Video	0.00703	25.14	GNVC-VD‡	0.0035	29.2
DCVC-RT	0.00668	31.29	DCVC-RT	0.00277	32.55
DCVC-FM	0.00589	31.09	DCVC-FM	0.00274	32.34
VTM	0.00575	29.95	HM	0.00278	31.18
HM	0.00599	29.19	x265	0.00284	30.08
x265	0.00520	27.75	VTM	0.00548	34.10
VoV+Scaling‡	0.0061	23.9	GLC-Video	0.00625	29.22
VoV‡	0.0059	17.9			

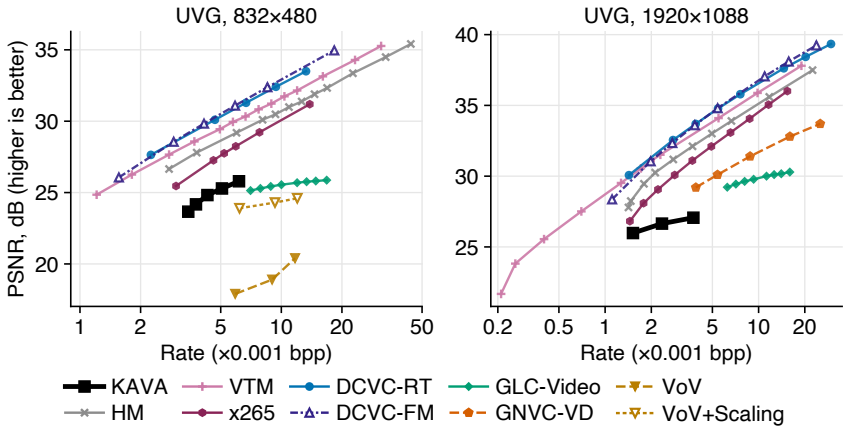

Figure 9: PSNR against rate on UVG at 832×480 (left) and 1920×1088 (right), seven-sequence means; higher is better, and rate is on a log scale. Frames and line styles as in Figure 5.

E TEMPORAL CONSISTENCY

Warp error is the masked mean squared difference between each frame and the previous frame warped along the optical flow of the video being scored, with flow from RAFT (Teed & Deng, 2020) (the large model with its released weights) and the forward-backward non-occlusion mask of Sundaram et al. (2010) (thresholds $\alpha_1 = 0.01$ and $\alpha_2 = 0.5$), in squared 8-bit pixel units; CLIP-F is the mean cosine similarity of consecutive frames' CLIP ViT-B/32 embeddings.

Neither measure ranks methods on its own: warp error falls when smoothing removes the detail that moves, so we read both together with LPIPS and DISTS, which smoothing hurts. Split by whether a frame pair crosses a clip boundary, our warp error is 49.13 against 11.87 for the other pairs (Section 4.1), and Figure 10 shows the rise at every boundary. The split compares two sets of frame pairs of the same video, so it does not depend on sharpness.

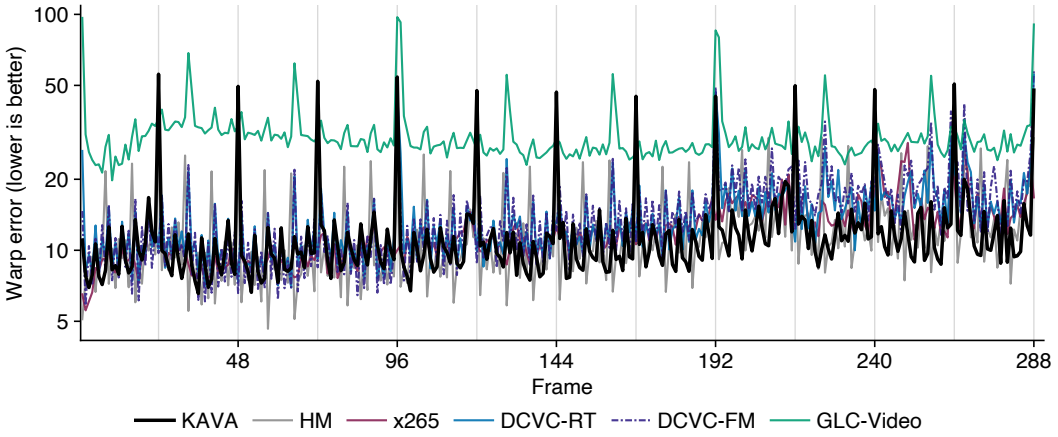

Figure 10: Warp error of each frame pair over the first 288 frames of UVG at 1920×1088, averaged over the seven sequences, with each method as in Table 3; log scale. The vertical lines mark our clip boundaries, every 24 frames, where two separately generated clips meet.

F EVALUATION ON THE FRAMES GNVC-VD AND VOV SCORE

We score our full-sequence reconstructions over frames 0 to 95 at 1920×1088 and 0 to 80 at 832×480, without refitting any adapter (Table 18). The rate is stated two ways. The whole-sequence rate divides everything sent for the sequence by all of its coded frames. It is what the prefix costs when the whole sequence is coded. The prefix-only rate charges the prefix with everything needed to decode it, the keyframes at positions 0 to 96 and the entire adapter. It is what coding the prefix on its own costs, and it matches the competitors' protocols. With the whole-sequence rate, our CRF 28 point is lower on both measures than GNVC-VD's curve at that rate. Our CRF 32 and CRF 35 points are lower on both measures than VoV+Scaling at its lowest published rate, by more than the reading error, although our CRF 32 point costs slightly more; at CRF 38 both differences are within it. With the prefix-only rate, our DISTS is 28 to 31 percent lower than GNVC-VD's at equal rate while our LPIPS is 0.011 to 0.024 higher. At 832×480 the prefix-only rate includes the adapter's 0.0170 bits per pixel, so every point of ours costs more than every published VoV point, and none of ours dominates either VoV configuration, even where our distance is lower. Over GNVC-VD's 96 frames the adapter alone costs 0.0027 bits per pixel.

Table 18: Our reconstructions scored on the frames GNVC-VD and VoV evaluate, UVG means. Competitor values are read from their published figures, one row per published point; their rates cover only the frames they score.

	Rate (bpp)			
Method	Whole seq.	Prefix only	LPIPS ↓	DISTS ↓
1920×1088, *frames* 0 *to* 95				
KAVA, CRF 28	0.00376	0.00794	0.2537	0.0621
KAVA, CRF 33	0.00235	0.00562	0.2779	0.0674
KAVA, CRF 38	0.00152	0.00436	0.3027	0.0752
GNVC-VD	0.0035		0.291	0.115
	0.0055		0.266	0.098
	0.0088		0.236	0.083
	0.0160		0.201	0.061
	0.0252		0.183	0.053
832×480, *frames* 0 *to* 80				
KAVA, CRF 32	0.00616	0.02467	0.2419	0.0859
KAVA, CRF 35	0.00507	0.02239	0.2580	0.0905
KAVA, CRF 38	0.00429	0.02073	0.2718	0.0952
KAVA, CRF 41	0.00377	0.01960	0.2873	0.0988
KAVA, CRF 44	0.00345	0.01889	0.3053	0.1090
VoV+Scaling	0.0061		0.275	0.096
	0.0091		0.258	0.088
	0.0120		0.250	0.081
VoV	0.0059		0.428	0.131
	0.0089		0.390	0.116
	0.0118		0.350	0.106

G Complexity and Cost

Table 19 lists the frozen components a receiver must hold. The unmerged adapter adds one rank-one product per adapted layer and step, which is negligible, so the codec runs at the speed of the foundation model. Decoding is one generation pass per clip, 50 Euler steps of the 16.40-billion-parameter transformer and one VAE decode, or 1,200 network evaluations for the 24 clips of a 600-frame sequence. Per frame at 832×480, encoding one operating point costs a median of 3.9 GPU-minutes and decoding 4.0 GPU-seconds. On UVG at 1920×1088 the medians are 9.9 GPU-minutes and 30.4 GPU-seconds. VoV reports, per frame, about 18 minutes of encoding on eight A100 GPUs in parallel (144 GPU-minutes) and 2.48 seconds of decoding.

Table 19: Frozen decoder-side components, measured from the released checkpoints. None is transmitted. The transformer count includes the image-conditioning projections, which the "14B" of the checkpoint's name leaves out.

Component	Parameters
Diffusion transformer	16,395,536,704
Variational autoencoder	126,892,531
Text encoder (uMT5)	5,680,910,336
Image encoder (CLIP)	1,193,014,785
Total, frozen	23,396,354,356
Transmitted adapter	81,360

H Qualitative Comparisons

Protocol. Figure 1 and Figures 11 to 14 use rates and encoder settings that differ from those of Table 1 and Figure 5. Each cell is labeled with its rate and the LPIPS distance of the region. Each competitor is shown at the cheapest of its settings at or above our rate, or at its lowest when none reaches it: x265 with its CRF searched per sequence, DCVC-RT and DCVC-FM with one intra



Figure 11: HEVC Class B at 1920×1088. GLC-Video is not shown. The bottom row enlarges the outlined part of the Cactus cells, a printed calendar, 5.4 times. In our cell the day numerals and weekday letters are legible and the saints' names are marks that resemble text; in DCVC-RT's cell the names are blurred away.

frame per sequence (81 frames at 832×480 and on HEVC Class B, the full sequence on UVG at 1920×1088), and GLC-Video at the lowest of its released rate points. DCVC-FM therefore appears at 2.8 to 25 times our printed rate. VoV* is our reduced-budget run of VoV's code (Appendix B).

Our cells and VoV's. Our cells show CRF 28 on UVG at 1920×1088, CRF 38 on HEVC Class B and CRF 44 at 832×480. Their printed rates charge the uncoded adapter and divide by source frames. The VoV* panels cover 81 frames and are enlarged from 832×480, so their distances include the enlargement; they are sharp but differ from the source in detail: the ReadySetGo gate plates read different numbers, and the Cactus calendar carries different numerals and letters.

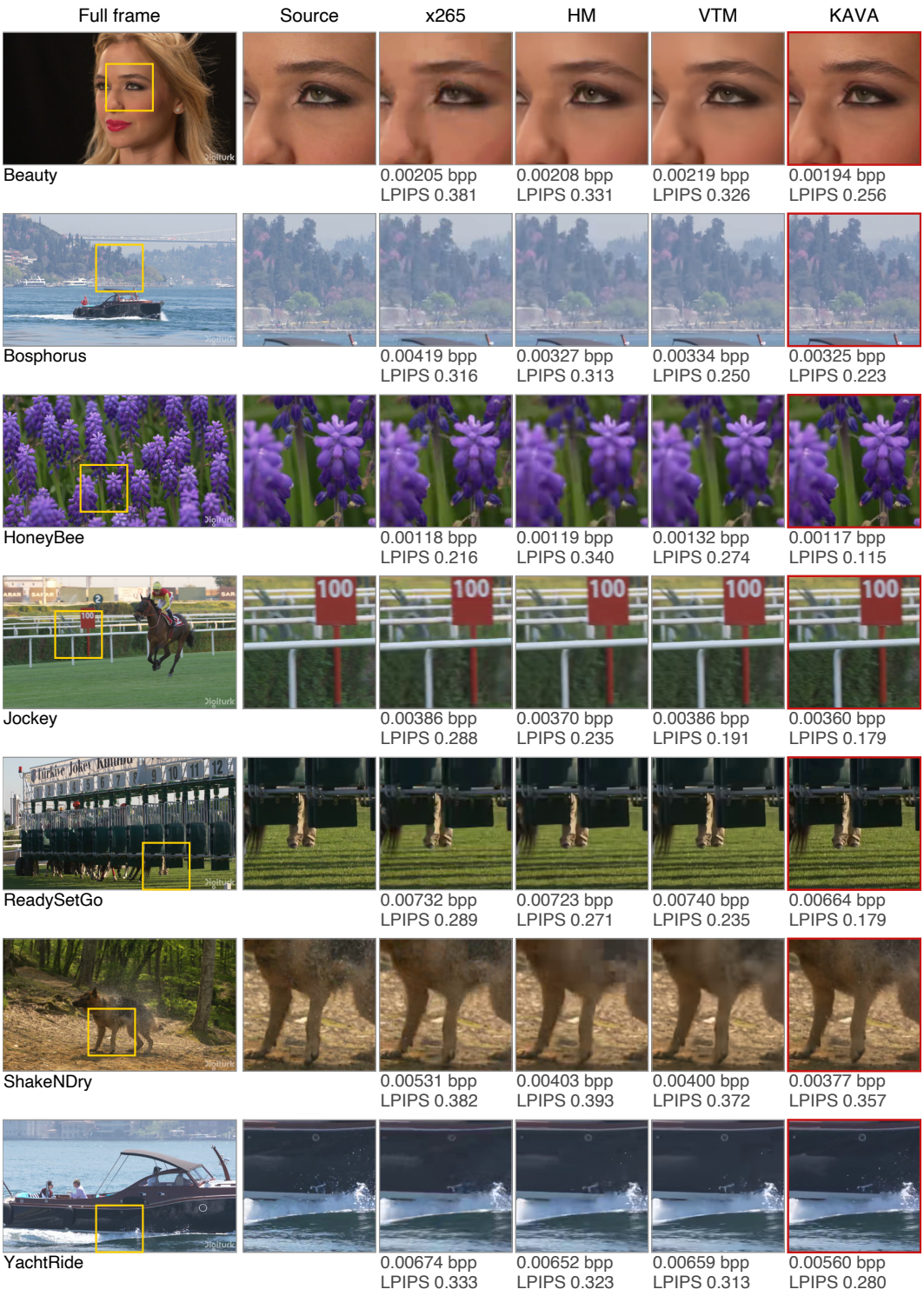

Figure 12: UVG at 1920×1088 against the conventional encoders. The florets in HoneyBee and the hedge behind the Jockey rail are blurred in the HM and VTM cells and textured in KAVA's.

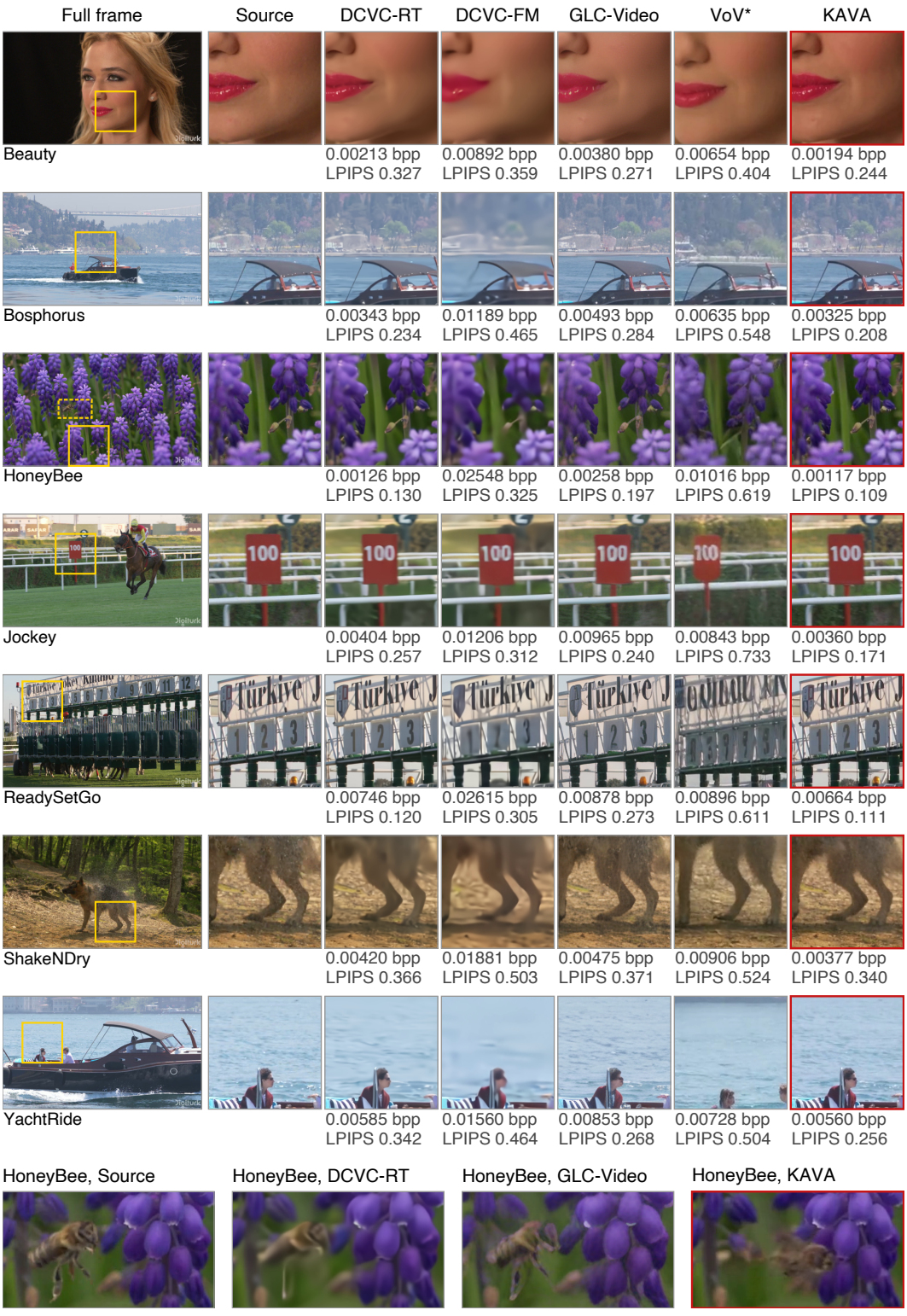

Figure 13: UVG at 1920 × 1088 against the learned and generative methods. HoneyBee's one moving subject, a small bee, is wrong: in the dashed box of its full frame, enlarged 2.3 times in the bottom row, our bee is smeared and sits right of the source's, DCVC-RT's is blurred, and GLC-Video's is sharp and at the source's position.

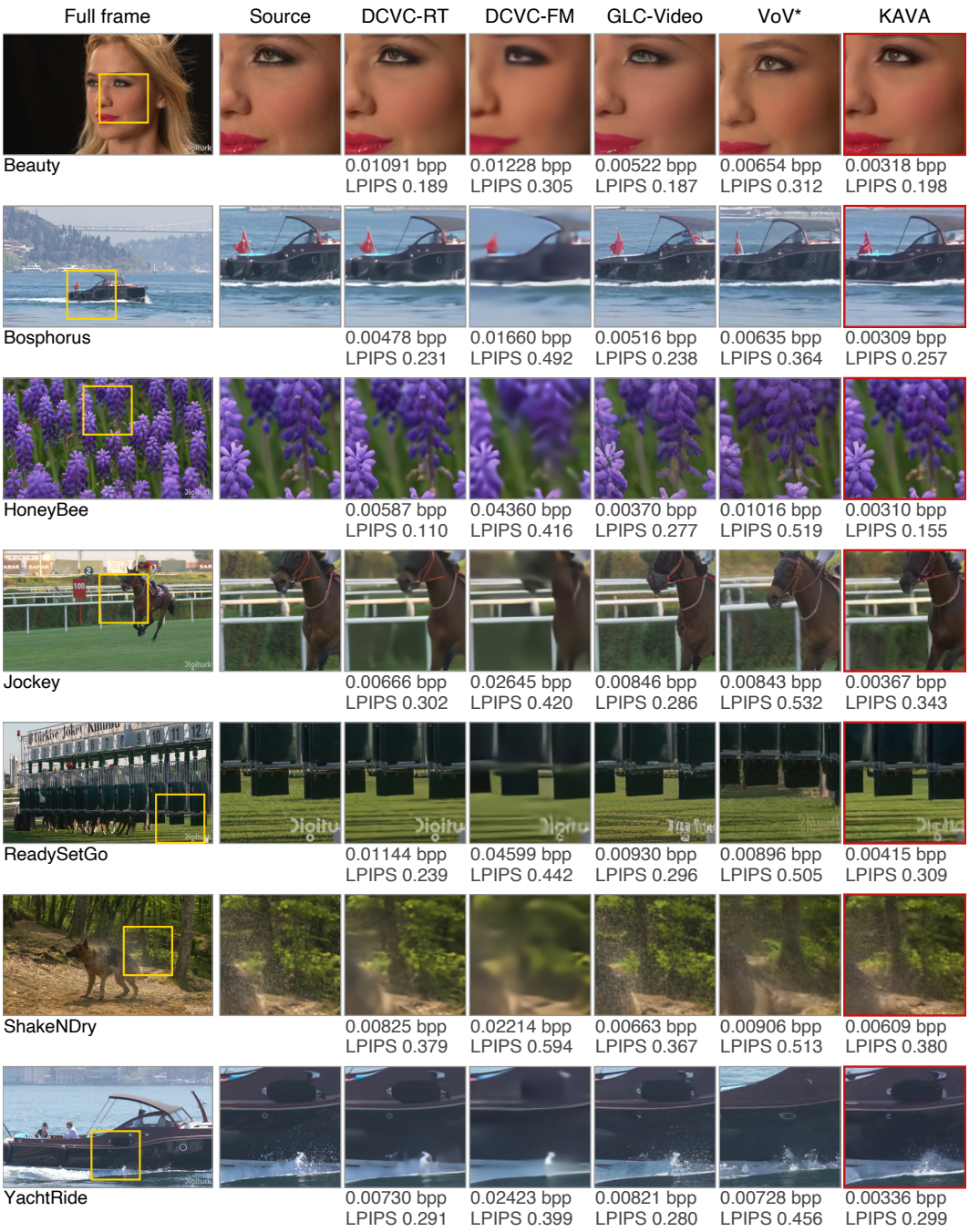

Figure 14: UVG at 832×480, CRF 44, against the learned and generative methods. Every competitor is at its lowest setting, which still costs more than our point. The printed distances of DCVC-RT are below ours on every region, and those of GLC-Video on all but HoneyBee.