RELIABLE AND PROPAGATION-AWARE TEST-TIME LATENT ADAPTATION FOR VIDEO COMPRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural video codecs rely on pretrained amortized encoders to generate coding latents, which may not be optimal for individual test videos. Encoder-side test-time latent adaptation provides an attractive way to improve instance-specific rate–distortion performance while keeping the pretrained codec and decoder unchanged. Extending latent refinement to chunk-based video coding, however, introduces two challenges: optimization is driven by differentiable surrogates that may not reflect native coding outcomes, and each adapted reconstruction changes the coding state used for subsequent chunks. We present, to our knowledge, the first encoder-side latent adaptation framework specifically designed for chunk-based neural video compression. At the chunk level, a gradient-conditioned learned optimizer generates coupled updates for the primary and hyper latents, while a distortion-constrained safety gate controls their local application with a fallback to the original encoder output. At the sequence level, candidate update policies are evaluated over a short observed prefix through separate causal closed-loop coding trajectories, using native bitstreams and decoded reconstructions to capture both actual coding behavior and temporal propagation effects. The selected policy is then causally applied from the state produced by the transmitted prefix, while all candidate trajectories remain private to the encoder. Our method requires no modification to the pretrained codec, decoder, or bitstream syntax, and introduces no additional signaling overhead. Experiments on six standard benchmark groups demonstrate consistent improvements over DCVC-UF, achieving an average BD-Rate reduction of 3.2% and up to 4.9% on individual datasets.

1 INTRODUCTION

Neural video compression has evolved from predictive coding (Lu et al., 2019; Rippel et al., 2019; Djelouah et al., 2019; Habibian et al., 2019) toward conditional coding (Li et al., 2021; 2022; Ho et al., 2022), with improved motion propagation and temporal-context modeling (Sheng et al., 2023; Mentzer et al., 2022; Qi et al., 2023; Li et al., 2023). Recent codecs further improve coding efficiency and long-range operation (Li et al., 2024; Tang et al., 2025; Jia et al., 2025), while DCVC-UF extends this direction to efficient chunk-based coding (Rippel et al., 2021; Li et al., 2024; Tang et al., 2025; Jia et al., 2025). Despite these advances, coding latents for all videos are still produced by a fixed pretrained encoder and may therefore be suboptimal for individual videos with diverse content and motion patterns. At test time, the encoder has access to the original input frames and can exploit instance-specific information to further refine the encoding. As illustrated in Fig.1, this asymmetry motivates encoder-side test-time adaptation, which exploits the original frames to improve rate–distortion performance while leaving the deployed decoder unchanged.

Test-time optimization has been widely explored in learned compression (Rozendaal et al., 2021; Wang et al., 2022a; Kim et al., 2024; Chen et al., 2025; Park et al., 2025), including neural video compression (Lu et al., 2020; Oh et al., 2024). Existing approaches adapt model parameters to individual inputs on the decoder-side components; the updated parameters or additional side information must also be communicated to the decoder, introducing extra transmission overhead. An alternative is to optimize the coding latents while keeping the pretrained codec entirely fixed. Such latent refinement has been shown to improve rate–distortion performance in learned image compression without modifying the decoding model (Campos et al., 2019; Yang et al., 2020b). Extending this strategy to a stateful neural video codec, however, is non-trivial, as each latent update affects not only the cur-

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.

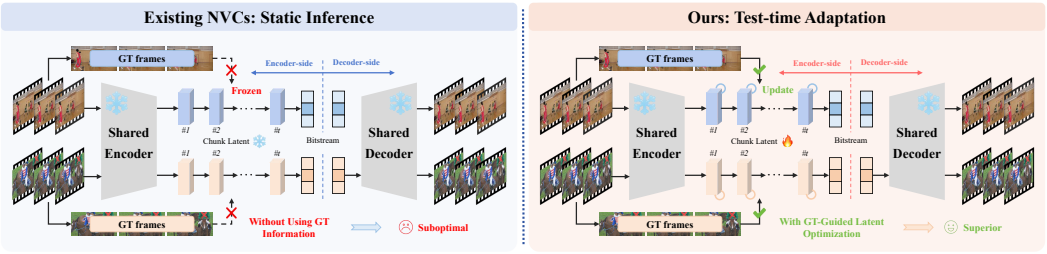


Figure 1: **Comparison between existing NVCs and our test-time adaptation framework.** Conventional chunk-based neural video codecs apply a fixed encoder to all test sequences, whereas our framework exploits the original frames available at the encoding side to adapt the coding latents while keeping the pretrained codec and decoding procedure unchanged.

rent chunk's rate–distortion outcome but also the coding state used for subsequent chunks, requiring adaptation to account for both local performance and subsequent coding trajectory.

This introduces two fundamental challenges. **CI:** local latent updates obtained by minimizing a differentiable rate–distortion objective are not uniformly reliable. An unconstrained update may excessively perturb the coding latents and degrade the reconstruction of the current chunk, particularly for unseen video content and operating conditions(Niu et al., 2022; Gong et al., 2022; Niu et al., 2023). **CII:** because each reconstructed chunk affects the coding state of subsequent chunks (Lu et al., 2020; Li et al., 2023; 2026), an update that is locally favorable to the current chunk may lead to unfavorable downstream effects. These two challenges motivate controlling the reliability of chunk-level adaptation while separately evaluating its utility under sequence-level state propagation.

We first develop reliability-controlled chunk-level latent adaptation. Inspired by learned optimization (Andrychowicz et al., 2016; Wichrowska et al., 2017; Metz et al., 2022), a gradient-conditioned optimizer generates coupled updates for the primary and hyper latents of each chunk, while a safety gate controls their magnitude using the current chunk's differentiable reconstruction distortion and falls back to the original encoder output when necessary. This separation lets the optimizer focus on generating useful update directions while the gate controls their local application. However, chunk-level control alone cannot establish whether an accepted update improves native rate–distortion performance or remains beneficial after propagating through subsequent coding states.

We then perform causal, propagation-aware sequence-level policy selection. Over a short observed prefix, the encoder evaluates candidate latent-update policies in separate closed-loop trajectories, each carrying its own decoded state, and scores them using accumulated native rate–distortion costs from actual bitstreams and decoded reconstructions. This captures both native coding behavior and the propagation effects of earlier adaptation decisions. Only one prefix bitstream is transmitted, while all candidate trajectories remain private to the encoder. After selection, the candidate states are discarded and the chosen policy is applied from the state produced by the transmitted prefix, allowing the decoder to remain synchronized without receiving optimizer parameters, candidate states, or a policy index. The prefix is used only to select the future encoding policy; transmitted chunks are not revised and no information from the unobserved suffix is used. Our contributions are summarized as follows:

- To the best of our knowledge, we propose the first test-time adaptation framework specifically designed for chunk-based neural video compression. We further present the first systematic study of encoder-side latent adaptation in this setting, jointly considering chunk-level adaptation reliability and sequence-level temporal propagation.
- We develop a gradient-conditioned latent adaptation method, together with a distortion-constrained safety gate that controls the local application of the proposed updates.
- We introduce a causal, propagation-aware policy selection procedure that evaluates candidate update policies over a short observed prefix using separate closed-loop coding trajectories and native rate–distortion outcomes.
- Experiments on standard video compression benchmarks demonstrate consistent rate–distortion improvements over the pretrained chunk-based codec, while keeping the codec parameters and decoding procedure unchanged without additional signaling overhead.

2 Related Work

Learned Video Compression. Early learned video codecs followed predictive coding by replacing handcrafted motion estimation, motion compensation, and residual coding with jointly optimized neural networks(Lu et al., 2019; Rippel et al., 2019; Djelouah et al., 2019; Habibian et al., 2019). Subsequent methods improved temporal prediction through scale-space motion, hierarchical quality structures, and feature-space representations(Agustsson et al., 2020; Yang et al., 2020a; Hu et al., 2021; Ho et al., 2022; Mentzer et al., 2022). Conditional coding further exploits temporal contexts for entropy modeling (Li et al., 2021; 2022; Sheng et al., 2023; Li et al., 2023), while recent work improves feature modulation and computational efficiency (Li et al., 2024; Rippel et al., 2021; Jia et al., 2025). These codecs nevertheless employ fixed inference procedures after training, leaving test-video information underexploited.

Test-Time Adaptation (TTA). TTA adapts pretrained models using information available during inference, including self-supervision, entropy minimization, continual adaptation, and augmentation-based objectives (Sun et al., 2020; Wang et al., 2021; 2022b; Gong et al., 2022; Zhang et al., 2022). In learned compression, instance-adaptive methods optimize coding latents or model parameters for individual inputs(Rozendaal et al., 2021; Oh et al., 2024; Chen et al., 2025). Existing approached primarily assess adaptation on individual image or coding units. In neural video compression, however, an adapted chunk also changes the decoded state of subsequent chunks. Our method therefore jointly considers chunk-level update reliability and sequence-level temporal propagation.

3 Method

A pretrained video encoder performs amortized inference, producing coding latents through a single feed-forward pass that may be suboptimal for a test sequence. We therefore refine these latents at test time while keeping the pretrained codec and decoder unchanged. In chunk-based video coding, however, refinement is stateful: each update affects both the current coding outcome and subsequent coding states. As illustrated in Fig.2, we address this with a two-level framework combining reliability-controlled chunk-level refinement and propagation-aware sequence-level policy selection.

3.1 Preliminaries

Consider a video consisting of an intra-coded first frame followed by T inter-coded chunks $\{\mathbf{x}_t\}_{t=1}^{T}$, where each chunk contains multiple consecutive frames. Let $\mathbf{s}_t$ denote the coding state available before encoding chunk t, which summarizes the information propagated from previously decoded chunks. Given the current chunk $\mathbf{x}_t$ and state $\mathbf{s}_t$, the pretrained encoder produces an initial primary latent $\mathbf{y}_t^{(0)}$ and hyper latent $\mathbf{z}_t^{(0)}$. After quantization and entropy coding, the resulting bitstream b_t is decoded to reconstruct the current chunk and update the coding state:

$$(\mathbf{y}_t^{(0)}, \mathbf{z}_t^{(0)}) = E_\phi(\mathbf{x}_t, \mathbf{s}_t), \qquad (\hat{\mathbf{x}}_t, \mathbf{s}_{t+1}) = D_\phi(b_t, \mathbf{s}_t). \tag{1}$$

Here, ϕ denotes the pretrained codec parameters, which remain fixed throughout our method. The intra-coded first frame is also left unchanged.

At test time, we refine the encoder-produced latents using the differentiable rate–distortion objective

$$\widetilde{\mathcal{J}}_t(\mathbf{y}, \mathbf{z}; \mathbf{s}_t) = \widetilde{R}_t(\mathbf{y}, \mathbf{z}; \mathbf{s}_t) + \lambda_q \widetilde{D}_t(\mathbf{x}_t, \widetilde{\mathbf{x}}_t), \tag{2}$$

where $\widetilde{R}_t$ denotes the differentiable rate estimate, $\widetilde{\mathbf{x}}_t$ is the corresponding differentiable reconstruction, and λ_q is the rate–distortion multiplier associated with QP q. We use R_t and D_t to denote the corresponding rate and distortion measured after native entropy coding and decoding.

The distinction between the differentiable objective and the native coding outcome is important in our setting. Moreover, since the decoded result of chunk t contributes to $\mathbf{s}_{t+1}$, a latent update can influence not only the current chunk but also subsequent coding decisions. These properties motivate the chunk-level adaptation and sequence-level policy selection described next.

3.2 Reliability-Controlled Chunk-level Latent Adaptation

We first perform latent adaptation at the individual chunk level. Rather than directly optimizing the latents with a fixed iterative optimizer at test time, we learn a gradient-conditioned update rule that

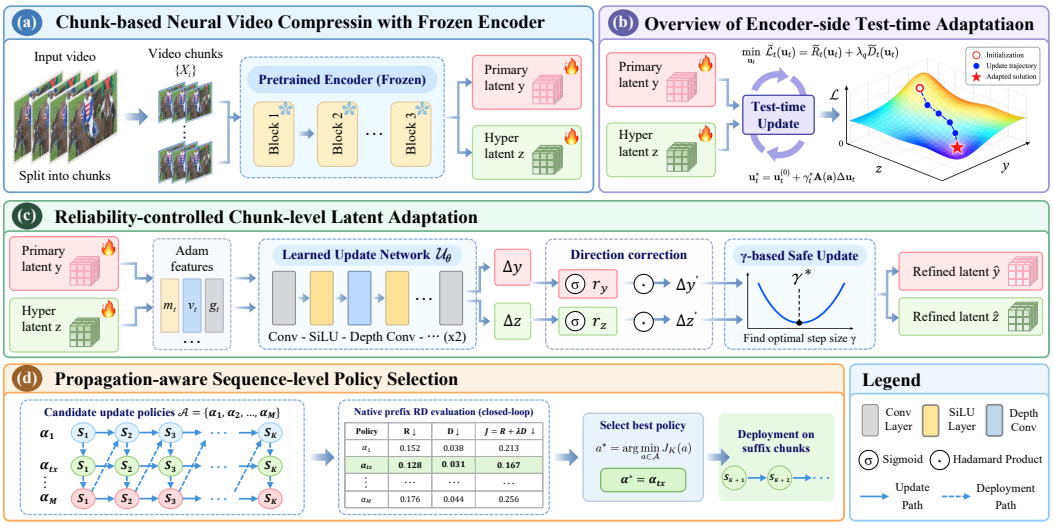


Figure 2: **Overview of the proposed encoder-side test-time adaptation framework.** (a) A frozen chunk-based neural video codec produces the primary and hyper latents for each chunk. (b) The encoder-side adaptation refines both latents toward a lower rate–distortion objective. (c) Reliability-controlled chunk-level adaptation generates gradient-conditioned dual-latent updates and determines their admissible magnitude through distortion-constrained safety control. (d) Propagation-aware sequence-level policy selection evaluates candidate update policies along private closed-loop prefix trajectories and causally commits the selected policy to subsequent chunks. All codec parameters and decoding-side operations remain unchanged.

predicts content-dependent refinement steps. Since these updates may not be reliable for every test chunk, a distortion-constrained safety gate controls their application.

Gradient-Conditioned Update Generation. Starting from the encoder prediction $(\mathbf{y}_t^{(0)}, \mathbf{z}_t^{(0)})$, we perform M steps of learned latent refinement. At step m, the frozen differentiable codec provides the gradients of the current-chunk rate–distortion objective:

$$(\mathbf{g}_{y,t}^{(m)}, \mathbf{g}_{z,t}^{(m)}) = \nabla_{(\mathbf{y},\mathbf{z})} \widetilde{\mathcal{J}}_t(\mathbf{y}_t^{(m)}, \mathbf{z}_t^{(m)}; \mathbf{s}_t). \tag{3}$$

We employ separate update modules for the primary and hyper latents. For either latent $v \in \{y, z\}$, let $\mathbf{h}_{v,t}^{(m)}$ collect its current value, gradient, first- and second-order Adam moments, and the normalized refinement step. The corresponding module uses these optimization states to modulate the standard Adam update and predict an additional residual correction:

$$\delta\mathbf{u}_t^{(m)} = G_{\theta_u}(\mathbf{h}_{u,t}^{(m)}) \odot \delta\mathbf{u}_{t,\mathrm{Adam}}^{(m)} + C_{\theta_u}(\mathbf{h}_{u,t}^{(m)}). \tag{4}$$

The latent is then updated as

$$\mathbf{u}_t^{(m+1)} = \mathbf{u}_t^{(m)} + \delta\mathbf{u}_t^{(m)}. \tag{5}$$

The update modules are initialized to reproduce the underlying Adam step and are subsequently trained to learn content-dependent modulation and correction. After M refinement steps, the optimizer produces a dual-latent displacement:

$$\boldsymbol{\delta}_t = (\delta\mathbf{y}_t, \delta\mathbf{z}_t) = \left(\mathbf{y}_t^{(M)} - \mathbf{y}_t^{(0)}, \mathbf{z}_t^{(M)} - \mathbf{z}_t^{(0)}\right). \tag{6}$$

This displacement represents a proposed latent update rather than an unconditional modification. Both the pretrained codec and learned update modules remain fixed throughout testing.

Distortion-Constrained Safety Gate. To control the reliability of a proposed update, we define a chunk-level gate for an arbitrary candidate displacement $\boldsymbol{\delta}_t = (\boldsymbol{\delta}_{y,t}, \boldsymbol{\delta}_{z,t})$. We scale it by $\gamma \in [0, 1]$ to form a one-dimensional path from the original encoder prediction toward the adapted latents:

$$\mathbf{u}_t(\gamma; \boldsymbol{\delta}_t) = (\mathbf{y}_t^{(0)}, \mathbf{z}_t^{(0)}) + \gamma\boldsymbol{\delta}_t. \tag{7}$$

Let $d_t(\gamma; \boldsymbol{\delta}_t)$ denote the mean per-frame reconstruction MSE obtained from the differentiable decoder along this path. Different from the complete rate–distortion objective in Eq. 2, d_t serves only as a local reliability criterion for screening potentially harmful updates. Given a tolerance $\epsilon \geq 0$, the full displacement is accepted when

$$d_t(1; \boldsymbol{\delta}_t) \leq (1 + \epsilon) d_t(0; \boldsymbol{\delta}_t). \tag{8}$$

If the full update violates Eq. 8, we search for a safer step along the same direction. Specifically, we construct a one-dimensional quadratic approximation using the distortion at the two endpoints and the directional derivative at the initial latents:

$$\widehat{d}_t(\gamma) = d_t(0) + u_t \gamma + c_t \gamma^2, \qquad c_t = d_t(1) - d_t(0) - u_t, \tag{9}$$

where

$$u_t = \left. \frac{\partial d_t(\gamma; \boldsymbol{\delta}_t)}{\partial \gamma} \right|_{\gamma=0}. \tag{10}$$

For readability, we omit the dependence on $\boldsymbol{\delta}_t$ in Eq. 9. A scalar search on the quadratic model obtains a candidate factor γ_t, which is verified using the differentiable codec. If the distortion constraint is violated, the update is rejected; otherwise, the scaled latents are passed to native entropy coding. The safety gate provides local control over the reconstruction behavior of each latent update. However, it neither directly constrains the native bitstream rate nor accounts for the effect of the adapted reconstruction on future coding states. We therefore further evaluate update policies at the sequence level in the following section.

3.3 Propagation-Aware Sequence-level Policy Selection

The chunk-level safety gate controls whether a proposed update is locally reliable, but it does not determine how strongly the update should be applied over an entire sequence. More importantly, different update strengths may induce different reconstructed states and therefore different downstream coding trajectories. To account for such temporal propagation, we evaluate a small set of candidate update policies over a short observed prefix and select the one with the best native rate–distortion performance for subsequent chunks.

Policy Scaling. We consider a finite set of candidate policies $\mathcal{A}$, where each policy $a = (\alpha_y^a, \alpha_z^a)$ specifies separate scaling factors for the primary and hyper-latent updates. Importantly, the learned displacement itself depends on the coding state from which it is generated. For a candidate policy with private state $\mathbf{s}_t^a$, let $\delta \mathbf{y}_t^a$ and $\delta \mathbf{z}_t^a$ denote the outputs of the chunk-level update generator under this state. The corresponding candidate displacement is

$$\boldsymbol{\delta}_t^a = \left(\alpha_y^a \delta \mathbf{y}_t^a,\, \alpha_z^a \delta \mathbf{z}_t^a \right). \tag{11}$$

The safety gate in Sec. 3.2 is then applied to $\boldsymbol{\delta}_t^a$, yielding a chunk-specific acceptance factor γ_t^a. Thus, (α_y^a, α_z^a) defines the overall update strength of a policy, while γ_t^a provides local reliability control conditioned on the current chunk and coding state.

Closed-Loop Candidate Trajectories. We evaluate the candidate policies over the first K inter-coded chunks. All candidates are initialized from the common state obtained after decoding the intra frame. Thereafter, each policy maintains its own private closed-loop state $\mathbf{s}_t^a$. When chunk $\mathbf{x}_t$ becomes available, candidate a generates its latent update from $(\mathbf{x}_t, \mathbf{s}_t^a)$, applies the policy scaling and safety gate, and performs native entropy coding and decoding:

$$(b_t^a, \hat{\mathbf{x}}_t^a, \mathbf{s}_{t+1}^a) = \mathcal{C}_\phi(\mathbf{x}_t, \mathbf{s}_t^a; a, \gamma_t^a), \tag{12}$$

where $\mathcal{C}_\phi$ denotes one complete native coding step under the fixed codec. Each subsequent prefix chunk is therefore evaluated from the state induced by the same candidate's preceding decisions. This closed-loop rollout is critical for capturing temporal propagation: evaluating all policies from a shared fixed state would reveal only their immediate chunk-level behavior while ignoring how earlier adaptation decisions affect later coding states.

Native Rate–Distortion Policy Evaluation. For each candidate policy, we accumulate its native rate–distortion cost over the observed prefix:

$$J_K(a) = \sum_{t=1}^{K} \left[R(b_t^a) + \lambda_q D(\mathbf{x}_t, \hat{\mathbf{x}}_t^a) \right], \qquad a^\star = \arg\min_{a \in \mathcal{A}} J_K(a). \tag{13}$$

Unlike the differentiable objective used for latent refinement, the rate in Eq. 13 is measured from the actual bitstream and the distortion from the native reconstruction. Since each candidate propagates its own coding state, $J_K(a)$ captures both immediate coding effects and their influence on later chunks. The evaluation is strictly causal: policy selection is performed after observing the first K chunks, without accessing frames from the unobserved suffix.

Prefix Transmission and Causal Commitment. Candidate evaluation is performed privately on the encoder side and is separate from actual bitstream transmission. During the prefix, only the bitstreams generated by a designated transmission policy a^{tx} are sent to the decoder. The remaining candidate bitstreams and coding states are used solely for encoder-side policy evaluation.

After selecting $a^\star$, we apply a fixed QP-aware acceptance heuristic based on its safety-gate activation ratio and native prefix-rate variation. If either statistic fails its corresponding threshold, the committed policy is conservatively replaced by the base policy $a^0 = (0, 0)$; otherwise, $a^\star$ is retained. All private candidate states are discarded, and the committed policy takes effect from chunk $K + 1$, starting from the coding state produced by the transmitted prefix:

$$\mathbf{s}_{K+1}^{\mathrm{enc}} = \mathbf{s}_{K+1}^{\mathrm{dec}} = \mathbf{s}_{K+1}^{\mathrm{tx}}. \tag{14}$$

When the base policy is committed, only suffix adaptation is disabled; the coding state produced by the transmitted prefix is retained, and no prefix chunk is replayed, revised, or re-encoded. The encoder therefore cannot continue from the private candidate state $\mathbf{s}_{K+1}^{a^\star}$, which is unavailable to the decoder. Encoder–decoder synchronization is preserved by continuing from the shared state in Eq. 14. Since policy selection only affects encoder-side latent generation, no policy index or candidate states need to be signaled. The prefix score serves only as an empirical criterion for future policy selection rather than an oracle estimate of suffix performance.

3.4 Training and Test-Time Deployment

Learning the Update Modules. We train the gradient-conditioned update modules offline while keeping all pretrained codec parameters fixed. To supervise the learned optimizer, we generate reference refinement trajectories by performing finite-step Adam optimization on Eq. 2. The overall training objective is defined as

$$\mathcal{L}_{\mathrm{train}} = \eta_{\mathrm{traj}} \mathcal{L}_{\mathrm{traj}} + \eta_{\mathrm{dir}} \mathcal{L}_{\mathrm{dir}} + \eta_{\mathrm{RD}} \mathcal{L}_{\mathrm{RD}}, \tag{15}$$

where $\mathcal{L}_{\mathrm{traj}}$ encourages the learned optimizer to imitate the finite-step refinement trajectory, $\mathcal{L}_{\mathrm{dir}}$ aligns the learned update direction with the reference optimization direction, and $\mathcal{L}_{\mathrm{RD}}$ supervises the final rate–distortion improvement of refined latents. The trajectory loss is normalized by the reference displacement scale to balance the primary and hyper latents with different numerical ranges. Detailed are provided in Appendix A.1. All reference trajectories are generated exclusively from offline training videos, and no optimized solution from a test sequence is used during training.

Encoder-Side Test-Time Adaptation. At test time, both the pretrained codec and learned update modules remain fixed. For each incoming chunk, the encoder generates a dual-latent displacement, applies the selected policy and safety gate, and performs native entropy coding. During the first K inter-coded chunks, the encoder additionally evaluates private closed-loop candidate trajectories and selects the future operating policy according to their native rate–distortion costs. The committed policy is then applied to subsequent chunks from the coding state produced by the transmitted prefix, without replaying or revising previous chunks. All adaptation operations are confined to the encoder. The decoder follows the standard bitstream decoding procedure and receives no adaptation information, enabling deployment without modifying the pretrained decoder or additional signaling.

Table 1: **BD-Rate (%) comparison on standard video compression benchmarks.** Results are computed in the YUV420 color space over all frames with the intra period set to -1. DCVC-UF serves as the anchor, and lower values indicate better compression performance.

Method	UVG	MCL-JCV	HEVC B	HEVC C	HEVC D	HEVC E	Average
HM-16.25	126.9	95.9	117.0	139.2	177.3	282.6	156.5
VTM-17.0	65.6	31.8	49.8	69.6	107.0	177.9	83.6
DCVC-DC	39.3	-2.9	21.0	0.2	9.0	409.1	79.3
DCVC-FM	34.5	21.8	24.2	13.8	23.9	103.7	37.0
DCVC-RT	23.8	6.9	20.1	21.7	36.1	134.4	40.5
DCVC-UF	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Ours	**-1.9**	**-3.2**	**-4.0**	**-4.7**	**-4.9**	**-0.9**	**-3.2**

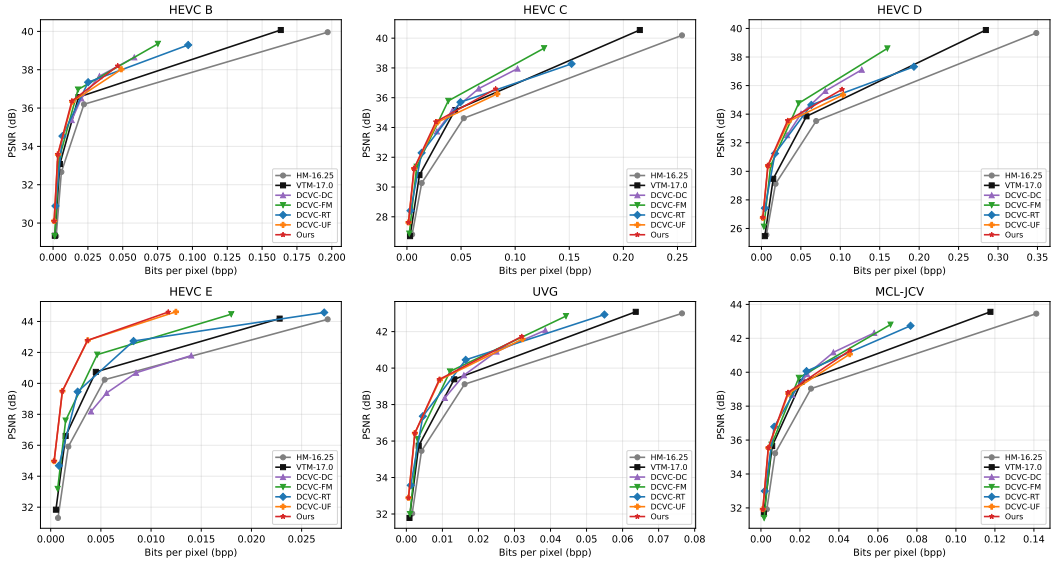


Figure 3: **Rate–distortion on six standard video compression benchmarks.** PSNR is measured in the YUV420 color space. Our method consistently shifts the DCVC-UF rate–distortion curves toward lower bitrates while retaining the pretrained codec and the original decoding procedure.

4 Experiments

4.1 Experimental Setup

Implementation Details. We use DCVC-UF as the frozen baseline codec. Unless otherwise specified, we perform ten latent-update steps per chunk and use $K = 8$. More implementation details are provided in Appendix A.2.

Training Details. We train one latent optimizer per rate point using 129-frame Vimeo sequences at 896×512. A 20-step Adam trajectory supervises the joint primary- and hyper-latent optimizer, which produces ten-step updates at inference. We set $\eta_{\text{traj}} = 1.0$, $\eta_{\text{dir}} = 0.05$, and $\eta_{\text{RD}} = 0.25$, while keeping the codec fixed.

Testing Details. We evaluate on the HEVC Class B$\sim$E (Bossen, 2013), UVG (Mercat et al., 2020), and MCL-JCV (Wang et al., 2016) using the weighted YUV420 PSNR over all frames with an intra period of -1. We report BD-Rate (Bjøntegaard, 2001) using native bitstreams. Our benchmarks include HM (Joint Collaborative Team on Video Coding, 2022) and VTM (Joint Video Experts Team, 2022) which represent the best H.265 and H.266 encoder, respectively. As for the NVC, we

compare with the recent SOTA including DCVC-DC (Li et al., 2023), DCVC-FM (Li et al., 2024), DCVC-RT (Jia et al., 2025), and DCVC-UF (Li et al., 2026).

4.2 Quantitative Results

Table 1 reports the BD-Rate results relative to DCVC-UF. Our method improves all six benchmark groups, with an average reduction of 3.2%. The largest gains occur on HEVC Class D, C, and B, reaching 4.9%, 4.7%, and 4.0%, respectively, followed by 3.2% on MCL-JCV. The smaller but consistent reductions on UVG and HEVC Class E indicate that the method remains conservative when the available adaptation gain is limited. The consistency is important because content-specific adaptation can otherwise benefit some sequences while degrading others.

Fig. 3 shows that the gains persist across the evaluated quality range rather than arising from a single rate point. The adapted curves generally shift toward lower bitrates at comparable reconstruction quality, with the clearest improvements on HEVC Class B-D and MCL-JCV. Together with table 1, these results show that the proposed adaptation produces stable coding gains across both content and quantization conditions. These gains require neither codec retraining nor decoder modification, showing that instance-specific latent inference complements architectural advances by recovering efficiency unavailable to amortized encoding. More results are provide in Appendix A.3.

4.3 Ablation Study

Component Analysis. Table 2 evaluates the contribution of each component. Compared with the marginal 0.094% gain of 20-step Adam refinement, the learned update achieves a 0.643% BD-Rate reduction, confirming that the improvement does not simply arise from additional optimization steps. Safety control further enlarges the reduction to 2.906%, showing that regulating update application is more consequential than update generation alone. Propagation-aware policy selection independently achieves 2.484%, confirming the importance of accounting for downstream coding states. Combining both mechanisms yields the full reduction of -3.241%. They operate at complementary scales: safety control regulates chunk-level update, whereas policy selection determines sequence-level use from native RD behavior over the prefix trajectory.

Table 2: **Component ablation averaged over six datasets.** Lower is better. $\mathcal{U}$, $\mathcal{S}$, and $\mathcal{P}$ denote the learned latent update, safety control, and propagation-aware policy selection, respectively.

Method	$\mathcal{U}$	$\mathcal{S}$	$\mathcal{P}$	Avg.
DCVC-UF				0.000
Adam refinement (20 steps)				-0.094
Learned update	✓			-0.643
Reliable chunk adaptation	✓	✓		-2.906
Sequence-level policy selection			✓	-2.484
Full method	✓	✓	✓	**-3.241**

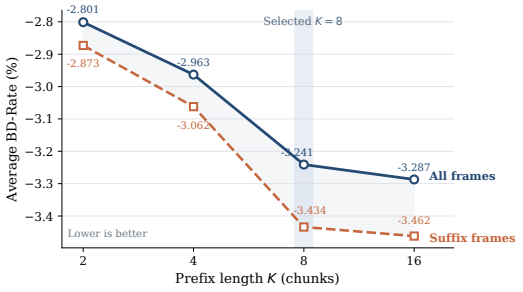


Figure 4: **Sensitivity to the prefix length K.** We report average BD-Rate over six datasets for all and suffix frames. Increasing K improves the reliability of sequence-level policy selection, while the gains largely saturate beyond $K = 8$.

Sensitivity to Prefix Length. Fig. 4 shows that increasing K from 2 to 8 improves the all-frame BD-Rate from -2.801% to -3.241%, as a longer observation window provides a more reliable estimate of candidate propagation behavior. The suffix BD-Rate similarly improves from -2.873% to -3.434%. Increasing K to 16 yields only 0.046 and 0.028 percentage points of additional gain, respectively, despite doubling the policy-evaluation horizon. We therefore use $K = 8$ to balance policy-selection reliability and prefix length.

4.4 Component-level Analysis

Chunk-level Reliability. Fig. 5(a) shows that the unconstrained learned update reduces the average BD-Rate by 0.643% but degrades UVG and HEVC Class E by 3.631% and 5.470%, respectively.

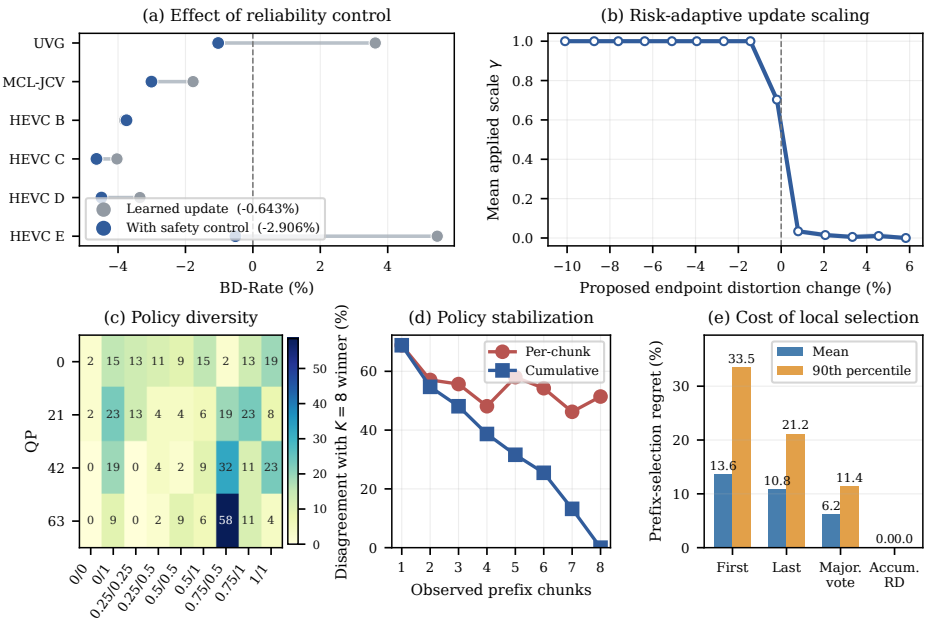


Figure 5: **Component-level analysis.** (a) Dataset-level BD-Rate obtained by the learned update before and after safety control. (b) Mean update scale γ as a function of the reconstruction-distortion change at the proposed update endpoint. Negative values indicate improved reconstruction. (c) Selection frequencies of candidate policies α_y/α_z across QPs. (d) Disagreement between policies selected from partial prefix evidence and the final policy selected using all $K = 8$ prefix chunks. (e) Selection regret of local criteria relative to accumulated native rate–distortion evaluation.

Safety control converts these degradations into reductions of 1.029% and 0.518% and improves the six-dataset average to 2.906%, demonstrating that the gate reduces the cross-risk of unconditional updates. Fig. 5(b) further explains this behavior: the gate preserves updates that reduce endpoint distortion, progressively scales them down near the reliability boundary, and suppresses harmful endpoints. It therefore retains beneficial directions while attenuating or rejecting updates toward unreliable region. Prefix accumulation therefore goes beyond smoothing local decisions by evaluating how early adaptation choices reshape subsequent coding states.

Sequence-level Selection. Reliable chunk-level application does not imply that a single scaling policy is optimal throughout the sequence. Fig. 5(c) shows that the preferred policy varies across content and QP. Selections are broadly distributed at lower QPs, whereas $(\alpha_y, \alpha_z) = (0.75, 0.5)$ is selected in 58% of the cases at QP 63. This diversity demonstrates the need to evaluate multiple policy-scaled trajectories rather than fixing one update magnitude globally.

Fig. 5(d) and 5(e) show that per-chunk choices disagree with the final $K = 8$ policy in over 46% of cases, whereas the cumulative winner stabilizes with additional observations. First chunk, last chunk, and majority voting incurs mean regrets of 13.6%, 10.8%, and 6.2%, with 90th-percentile regrets of 33.5%, 21.2%, and 11.4%, respectively. Accumulated native rate–distortion evaluation thus reduces local-selection instability and complements safety control by determining sequence-level update use. Because this comparison uses native rate and distortion, the selected policy reflects actual coding outcomes rather than local surrogate improvements. This distinction explains why local reliability and sequence-level utility require separate controls.

5 Conclusion

We presented an encoder-side test-time adaptation framework for chunk-based neural video compression. By combining gradient-conditioned dual-latent updates, chunk-level reliability control, and propagation-aware policy selection, our method improves the coding latents while accounting for both local adaptation reliability and temporal state propagation. Experiments on six benchmark groups demonstrate consistent bitrate reductions over a strong pretrained codec, with no modification to the codec parameters, decoding procedure, or transmitted syntax.

AI USE STATEMENT

(This section is **required** and does not count toward the page limit.)

In this work, we used generative AI tools solely for language polishing of the manuscript. We did not use generative AI tools for ideation, research methodology, experimental design, data analysis, or result generation. All research concepts, ideas, analyses, and experiments were developed and conducted by the authors. The use of generative AI was limited to improving the linguistic quality of the manuscript, with no involvement in the scientific content or data analysis. All AI-assisted text was reviewed, edited, and approved by the authors. We take full responsibility for the final content of this work, including any text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

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

Marcin Andrychowicz, Misha Denil, Sergio Gómez, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

Gisle Bjøntegaard. Calculation of average PSNR differences between RD curves. Technical Report VCEG-M33, Video Coding Experts Group, 2001.

Frank Bossen. Common test conditions and software reference configurations. Technical Report JCTVC-L1100, Joint Collaborative Team on Video Coding, 2013.

Joaquim Campos, Simon Meierhans, Abdelaziz Djelouah, and Christopher Schroers. Content adaptive optimization for neural image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

Kecheng Chen, Pingping Zhang, Tiexin Qin, Shiqi Wang, Hong Yan, and Haoliang Li. Test-time adaptation for image compression with distribution regularization. In *International Conference on Learning Representations*, 2025.

Abdelaziz Djelouah, Joaquim Campos, Simone Schaub-Meyer, and Christopher Schroers. Neural inter-frame compression for video coding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6421–6429, 2019.

Taesik Gong, Jongheon Jeong, Taewon Kim, Yewon Kim, Jinwoo Shin, and Sung-Ju Lee. NOTE: Robust continual test-time adaptation against temporal correlation. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

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

Yung-Han Ho, Chih-Peng Chang, Peng-Yu Chen, Alessandro Gnutti, and Wen-Hsiao Peng. CANF-VC: Conditional augmented normalizing flows for video compression. In *European Conference on Computer Vision*, pp. 207–223, 2022.

Zhihao Hu, Guo Lu, and Dong Xu. FVC: A new framework towards deep video compression in feature space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1502–1511, 2021.

Zhaoyang Jia, Bin Li, Jiahao Li, Wenxuan Xie, Linfeng Qi, Houqiang Li, and Yan Lu. Towards practical real-time neural video compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12543–12552, 2025.

Joint Collaborative Team on Video Coding. HM-16.25: HEVC test model reference software, 2022. Version 16.25.

Joint Video Experts Team. VTM-17.0: VVC test model reference software, 2022. Version 17.0.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9347–9358, 2024.

Jiahao Li, Bin Li, and Yan Lu. Deep contextual video compression. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Jiahao Li, Bin Li, and Yan Lu. Hybrid spatial-temporal entropy modelling for neural video compression. In *Proceedings of the 30th ACM International Conference on Multimedia*, pp. 1503–1511, 2022.

Jiahao Li, Bin Li, and Yan Lu. Neural video compression with diverse contexts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22616–22626, 2023.

Jiahao Li, Bin Li, and Yan Lu. Neural video compression with feature modulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 26099–26108, 2024.

Jiahao Li, Wenxuan Xie, Zhaoyang Jia, Bin Li, Zongyu Guo, Xiaoyi Zhang, and Yan Lu. Ultra-fast neural video compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. DVC: An end-to-end deep video compression framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11006–11015, 2019.

Guo Lu, Chunlei Cai, Xiaoyun Zhang, Li Chen, Wanli Ouyang, Dong Xu, and Zhiyong Gao. Content adaptive and error propagation aware deep video compression. In *European Conference on Computer Vision*, 2020.

Fabian Mentzer, George D. Toderici, David Minnen, Sergi Caelles, Sung Jin Hwang, Mario Lucic, and Eirikur Agustsson. VCT: A video compression transformer. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

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*, pp. 297–302, 2020.

Luke Metz, C. Daniel Freeman, James Harrison, Niru Maheswaranathan, and Jascha Sohl-Dickstein. Practical tradeoffs between memory, compute, and performance in learned optimizers. In *Proceedings of the First Conference on Lifelong Learning Agents*, pp. 142–164, 2022.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 16888–16905, 2022.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Towards stable test-time adaptation in dynamic wild world. In *International Conference on Learning Representations*, 2023.

Seungjun Oh, Hyunmo Yang, and Eunbyung Park. Parameter-efficient instance-adaptive neural video compression. In *Proceedings of the Asian Conference on Computer Vision*, 2024.

Unki Park, Seongmoon Jeong, Youngchan Jang, Gyeong-Moon Park, and Jong Hwan Ko. Test-time fine-tuning of image compression models for multi-task adaptability. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

Linfeng Qi, Jiahao Li, Bin Li, Houqiang Li, and Yan Lu. Motion information propagation for neural video compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

Oren Rippel, Sanjay Nair, Carissa Lew, Steve Branson, Alexander G. Anderson, and Lubomir Bourdev. Learned video compression. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3454–3463, 2019.

Oren Rippel, Alexander G. Anderson, Kedar Tatwawadi, Sanjay Nair, Craig Lytle, and Lubomir Bourdev. ELF-VC: Efficient learned flexible-rate video coding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 14479–14488, 2021.

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*, 2021.

Xihua Sheng, Jiahao Li, Bin Li, Li Li, Dong Liu, and Yan Lu. Temporal context mining for learned video compression. *IEEE Transactions on Multimedia*, 25:7311–7322, 2023.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 9229–9248, 2020.

Chuanbo Tang, Zhuoyuan Li, Yifan Bian, Li Li, and Dong Liu. Neural video compression with context modulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12553–12563, 2025.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021.

Dezhao Wang, Wenhan Yang, Yueyu Hu, and Jiaying Liu. Neural data-dependent transform for learned image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022a.

Haiqiang Wang, Weihao Gan, Sudeng Hu, Joe Yuchieh Lin, Lina Jin, Longguang Song, Ping Wang, Ioannis Katsavounidis, Anne Aaron, and C.-C. Jay Kuo. MCL-JCV: A JND-based H.264/AVC video quality assessment dataset. In *IEEE International Conference on Image Processing*, pp. 1509–1513, 2016.

Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7201–7211, 2022b.

Olga Wichrowska, Niru Maheswaranathan, Matthew W. Hoffman, Sergio Gómez Colmenarejo, Misha Denil, Nando de Freitas, and Jascha Sohl-Dickstein. Learned optimizers that scale and generalize. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 3751–3760, 2017.

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

Yibo Yang, Robert Bamler, and Stephan Mandt. Improving inference for neural image compression. In *Advances in Neural Information Processing Systems*, volume 33, 2020b.

Marvin Zhang, Sergey Levine, and Chelsea Finn. MEMO: Test time robustness via adaptation and augmentation. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

A Appendix

A.1 Training-Loss Definitions

We provide the detailed definitions of the losses used to train the gradient-conditioned update modules. For each training chunk, a 20-step Adam refinement is performed on the differentiable rate–distortion objective in Eq. 2. The resulting trajectory is sampled every two steps to produce $M = 10$

reference states, denoted by $\{\bar{\mathbf{v}}_t^{(m)}\}_{m=1}^M$ for each latent $\mathbf{v} \in \{\mathbf{y}, \mathbf{z}\}$. These states are temporally aligned with the M states $\{\mathbf{v}_t^{(m)}\}_{m=1}^M$ generated by the learned optimizer.

To account for the different numerical scales of the primary and hyper latents, we normalize each reference displacement by its root-mean-square magnitude:

$$s_{\mathbf{v},t}^{(m)} = \max\left(\sqrt{\frac{1}{N_{\mathbf{v}}}\left\|\bar{\mathbf{v}}_t^{(m)} - \mathbf{v}_t^{(0)}\right\|_2^2}, \epsilon\right), \qquad \mathbf{v} \in \{\mathbf{y}, \mathbf{z}\}, \tag{16}$$

where $N_{\mathbf{v}}$ is the number of elements in $\mathbf{v}$ and $\epsilon = 10^{-6}$. The trajectory-matching loss is then defined as

$$\mathcal{L}_{\text{traj}} = \frac{1}{2M} \sum_{\mathbf{v}\in\{\mathbf{y},\mathbf{z}\}} \sum_{m=1}^{M} \text{SmoothL1}\left(\frac{\mathbf{v}_t^{(m)} - \mathbf{v}_t^{(0)}}{s_{\mathbf{v},t}^{(m)}}, \frac{\bar{\mathbf{v}}_t^{(m)} - \mathbf{v}_t^{(0)}}{s_{\mathbf{v},t}^{(m)}}\right). \tag{17}$$

This loss encourages the learned optimizer to reproduce the intermediate finite-step refinement dynamics rather than only matching the final solution.

We additionally align the direction of each learned displacement with its reference counterpart:

$$\mathcal{L}_{\text{dir}} = \frac{1}{2M} \sum_{\mathbf{v}\in\{\mathbf{y},\mathbf{z}\}} \sum_{m=1}^{M} \left[1 - \frac{\left\langle \mathbf{v}_t^{(m)} - \mathbf{v}_t^{(0)}, \bar{\mathbf{v}}_t^{(m)} - \mathbf{v}_t^{(0)} \right\rangle}{\left\|\mathbf{v}_t^{(m)} - \mathbf{v}_t^{(0)}\right\|_2 \left\|\bar{\mathbf{v}}_t^{(m)} - \mathbf{v}_t^{(0)}\right\|_2 + \epsilon_{\cos}}\right], \tag{18}$$

where $\epsilon_{\cos} = 10^{-8}$ ensures numerical stability. Finally, the rate–distortion term directly evaluates the final learned latents relative to the initial encoder prediction:

$$\mathcal{L}_{\text{RD}} = \frac{\widetilde{\mathcal{J}}_t\left(\mathbf{y}_t^{(M)}, \mathbf{z}_t^{(M)}; \mathbf{s}_t\right)}{\widetilde{\mathcal{J}}_t\left(\mathbf{y}_t^{(0)}, \mathbf{z}_t^{(0)}; \mathbf{s}_t\right)}. \tag{19}$$

The complete training objective is

$$\mathcal{L}_{\text{train}} = \eta_{\text{traj}}\mathcal{L}_{\text{traj}} + \eta_{\text{dir}}\mathcal{L}_{\text{dir}} + \eta_{\text{RD}}\mathcal{L}_{\text{RD}}, \tag{20}$$

with $\eta_{\text{traj}} = 1.0$, $\eta_{\text{dir}} = 0.05$, and $\eta_{\text{RD}} = 0.25$. The first two terms transfer the finite-step optimization dynamics and update directions from the Adam reference, while the final term directly encourages the learned trajectory to terminate at latents with a lower differentiable rate–distortion cost.

A.2 Implementation Details

Update Modules. We employ separate update modules for the primary latent y and hyper latent z. At optimization step m, each module receives the standardized current latent, rate–distortion gradient, bias-corrected first-order Adam moment, square root of the bias-corrected second-order moment, and normalized step index. For $v \in \{y, z\}$, the input feature is

$$\phi_v^{(m)} = \text{Concat}\left[\mathcal{N}\left(v^{(m-1)}\right), \mathcal{N}\left(g_v^{(m)}\right), \mathcal{N}\left(\widehat{m}_v^{(m)}\right), \mathcal{N}\left(\sqrt{\widehat{s}_v^{(m)}}\right), \frac{m-1}{M-1}\mathbf{1}\right], \tag{21}$$

where $\mathcal{N}(\cdot)$ denotes feature standardization. Each update module consists of alternating pointwise and depthwise convolutions with SiLU activations. Its two output heads predict a multiplicative gate and an additive correction:

$$\Delta v^{(m)} = G_{\theta_v}\left(\phi_v^{(m)}\right) \odot \Delta v_{\text{Adam}}^{(m)} + C_{\theta_v}\left(\phi_v^{(m)}\right), \tag{22}$$

where

$$G_{\theta_v} = g_{\max}\sigma\left(\widetilde{G}_{\theta_v}\right), \qquad C_{\theta_v} = \rho\ \text{RMS}\left(\Delta v_{\text{Adam}}^{(m)}\right)\tanh\left(\widetilde{C}_{\theta_v}\right). \tag{23}$$

The hidden dimensions of the y- and z-update modules are 96 and 64, respectively. We set $g_{\max} = 2.5$ and $\rho = 1$. The final layers are initialized such that the initial update reproduces the underlying Adam step, providing a stable starting point for offline training.

Table 3: **Per-sequence BD-Rate (%) relative to DCVC-UF.** All and Suffix denote results over all frames and suffix frames, respectively. Lower is better.

Dataset	Sequence	All	Suffix
UVG	Beauty	-1.481	-1.615
	Bosphorus	-0.107	-0.275
	HoneyBee	1.623	0.595
	Jockey	-4.621	-4.746
	ReadySteadyGo	-2.023	-1.833
	ShakeNDry	0.836	0.294
	YachtRide	-1.905	-1.798
MCL-JCV	SRC01	6.387	2.093
	SRC02	-5.184	-5.965
	SRC03	-2.258	-3.407
	SRC04	-2.481	-3.556
	SRC05	-9.385	-10.473
	SRC06	4.206	2.485
	SRC07	-4.461	-3.431
	SRC08	-2.119	-2.748
	SRC09	-1.499	-1.561
	SRC10	-7.016	-7.100
	SRC11	-5.816	-6.234
	SRC12	0.826	0.513
	SRC13	0.130	0.029
	SRC14	-4.456	-5.404
	SRC15	-0.346	-0.847
	SRC16	0.784	-2.518
	SRC17	-2.496	-3.182
	SRC18	-0.927	3.351
	SRC19	-5.240	-5.626
	SRC20	-3.750	-3.459
MCL-JCV	SRC21	-4.193	-5.382
	SRC22	-4.744	-4.556
	SRC23	-2.779	-3.089
	SRC24	-7.368	-7.103
	SRC25	-4.441	-5.498
	SRC26	-5.385	-5.868
	SRC27	-1.633	-1.944
	SRC28	1.681	1.337
	SRC29	10.929	1.504
	SRC30	1.969	0.819
HEVC B	BasketballDrive	-8.066	-8.207
	BQTerrace	-3.530	-3.462
	Cactus	-1.570	-1.625
	Kimono1	-2.099	-1.965
	ParkScene	-1.289	-1.277
HEVC C	BasketballDrill	-4.993	-4.794
	BQMall	-4.814	-4.575
	PartyScene	-2.469	-2.403
	RaceHorses	-9.888	-9.991
HEVC D	BasketballPass	-6.954	-7.188
	BQSquare	-2.691	-2.883
	BlowingBubbles	-2.104	-1.892
	RaceHorses	-8.811	-8.865
HEVC E	FourPeople	-0.446	-0.492
	Johnny	-1.238	-1.417
	KristenAndSara	-1.105	-1.363

Optimization Settings. The learned optimizer performs $M = 10$ update steps. The underlying Adam updates use learning rates of 10^{-2} and 3×10^{-3} for y and z, respectively, with moment coefficients $(0.9, 0.999)$. The joint gradient norm of the two latents is clipped to 1. The update modules are trained using Adam with an outer learning rate of 2×10^{-4}. Reference trajectories are generated by 20-step Adam refinement, with every two reference steps corresponding to one learned update step. Training uses 256×256 video patches.

Reliability Control. For an initial latent pair $\mathbf{u}_t^{(0)} = (y_t^{(0)}, z_t^{(0)})$ and a policy-scaled update direction $\Delta\mathbf{u}_t$, we determine a scalar $\gamma_t^\star \in [0, 1]$ such that the differentiable reconstruction distortion does not exceed that of the initial encoder prediction. Let

$$d_t(\gamma; \Delta\mathbf{u}_t) = \frac{1}{F_t} \sum_{i=1}^{F_t} \mathrm{MSE}\left(x_{t,i}, \widehat{x}_{t,i}\left(\mathbf{u}_t^{(0)} + \gamma\Delta\mathbf{u}_t\right)\right) \tag{24}$$

denote the mean per-frame reconstruction MSE of the current chunk. We approximate the distortion variation along the update direction by

$$d_t(\gamma; \Delta\mathbf{u}_t) - d_t(0) \approx s_t\gamma + c_t\gamma^2, \tag{25}$$

where

$$s_t = \langle \nabla_{\mathbf{u}} d_t(0), \Delta\mathbf{u}_t \rangle, \qquad c_t = d_t(1; \Delta\mathbf{u}_t) - d_t(0) - s_t. \tag{26}$$

If the full update satisfies the distortion constraint, we set $\gamma_t^\star = 1$. Otherwise, we use bisection on the fitted scalar quadratic to obtain the largest feasible scale:

$$\gamma_t^\star = \max_{\gamma \in [0,1]} \left\{\gamma : s_t\gamma + c_t\gamma^2 \le \epsilon_t d_t(0)\right\}, \tag{27}$$

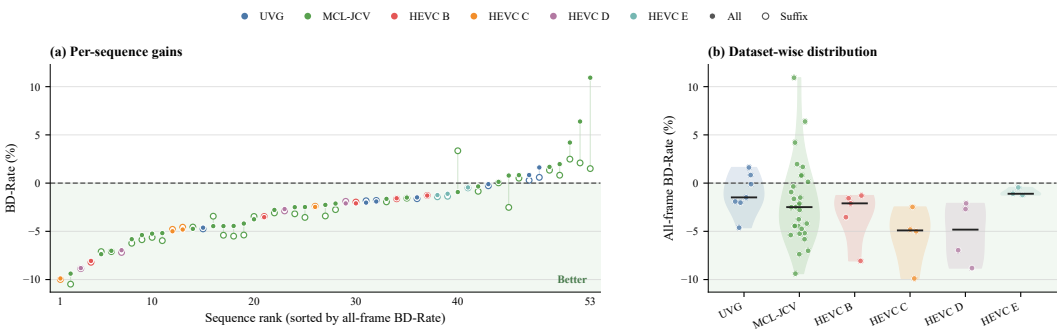


Figure 6: **Per-sequence BD-Rate results.** (a) All-frame and suffix results for 53 sequences, sorted by all-frame BD-Rate. (b) Dataset-wise all-frame distributions; horizontal bars denote medians. DCVC-UF is the anchor, and lower values are better.

where $\epsilon_t = 0$ in our experiments. Because the quadratic model is an approximation, the selected update is verified once using the differentiable decoder. It is rejected by setting $\gamma_t^\star = 0$ if the distortion constraint is violated.

Sequence-level Policy Selection. We consider the candidate scaling policies

$$\begin{aligned}\mathcal{A} = \{&(0,0),\ (0,1),\ (0.25,0.25),\ (0.25,0.5),\\ &(0.5,0.5),\ (0.5,1),\ (0.75,0.5),\ (0.75,1),\ (1,1)\},\end{aligned} \tag{28}$$

where the two entries scale the reliability-controlled updates of y and z, respectively. Each policy maintains an independent causal coding state over the first $K = 8$ chunks. Its prefix score is computed from the accumulated native rate–distortion cost:

$$J_K(\boldsymbol{\alpha}) = \sum_{t=1}^{K} \left[R_t^{\mathrm{nat}}(\boldsymbol{\alpha}) + \lambda D_t^{\mathrm{nat}}(\boldsymbol{\alpha})\right], \qquad \widehat{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\alpha}\in\mathcal{A}} J_K(\boldsymbol{\alpha}), \tag{29}$$

where R_t^{nat} is measured from the produced bitstream and D_t^{nat} is computed from the native decoded reconstruction.

QP-aware Acceptance. The prefix-selected policy is committed only if its observed behavior satisfies the QP-aware acceptance rule. For QPs 0, 21, 42, and 63, the minimum safety-gate activation ratios are 0.25, 0, 0, and 0.5, respectively, while the corresponding maximum mean native prefix-rate increases are 1%, 0%, 0%, and 3%. These empirically specified thresholds are fixed throughout evaluation. If either condition is violated, the sequence continues with the original encoder policy $(0, 0)$. Only the original-policy prefix is transmitted; candidate prefix trajectories are used solely for causal policy evaluation and are discarded after selection. The selected policy is then applied to the suffix from the transmitted prefix state, leaving the decoder and bitstream syntax unchanged.

A.3 Per-Sequence Results

Table 3 reports the per-sequence BD-Rate relative to DCVC-UF. Our method reduces the all-frame BD-Rate on 42 of the 53 test sequences and achieves particularly pronounced gains on BasketballDrive, RaceHorses, videoSRC05, and videoSRC24. The gains are distributed across datasets rather than being dominated by a small number of sequences. Moreover, the suffix results are generally consistent with the all-frame results, confirming that the selected policy remains effective after the prefix used for policy evaluation.

Fig. 6 further summarizes the sequence-level behavior. Most sequences lie below the zero-BD-Rate boundary, with particularly consistent gains on HEVC Class B–E. The close agreement between the all-frame and suffix results indicates that the selected policies generally remain effective beyond the prefix used for policy evaluation. Larger variations are observed on MCL-JCV, which contains both the strongest improvements and several failure cases. Nevertheless, its distribution remains concentrated on the negative side, and all six dataset-level averages favor the proposed method.