PAIRNeRV: Patch-Owned Appearance and Innovation for Neural Video Representation

Anonymous authors
Paper under double-blind review

Abstract

Neural video representations offer compact reconstruction, but fitting them separately to each video incurs substantial encoding cost. Feed-forward token-based methods avoid this optimization by inferring video codes for a shared decoder. However, using a global token memory leaves temporal sharing and spatial relevance implicit in the stored representation, while reconstruction repeatedly retrieves content through attention. We propose **PAIRNeRV**, a feed-forward framework that organizes video state by temporal role and spatial ownership. A clip-wide encoder gathers global context, from which two writers produce appearance codes shared across frames and innovation codes specific to individual frames, both assigned to spatial patches. A shared decoder reconstructs requested patches through direct lookup of their appearance and innovation codes, without global retrieval attention. Appearance and geometry projections are reused across frames. Eight-bit storage complements the design by retaining more code values within a fixed bit budget. Experiments show improved reconstruction over the compared feed-forward baselines under a common video-state budget and transfer to unseen datasets without fine-tuning. Comparisons with global-attention readout demonstrate faster resident-state decoding, while controlled studies examine the roles of state organization, storage allocation, and cross-frame reuse.

1 Introduction

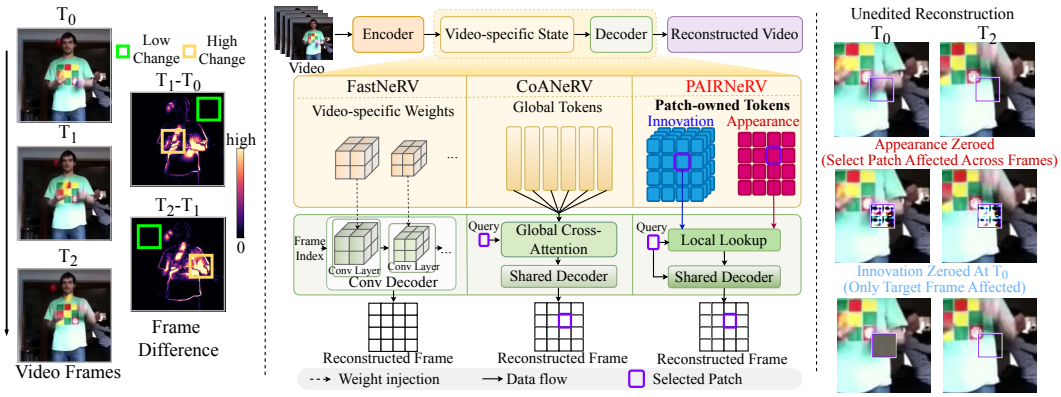
Video representation must preserve spatial detail and temporal variation within a limited storage budget while allowing efficient reconstruction. Neural video representations learn functions that map frame indices or coordinates to image values. Neural Representations for Videos (NeRV) (Chen et al., 2021a) stores a video's content in network weights, and subsequent methods (Li et al., 2022; Chen et al., 2023; Kwan et al., 2023) improve reconstruction through neural architectures and associated features fitted to each video. Decoding is forward inference once the representation is fitted, but encoding each new video still requires costly optimization.

Feed-forward encoding addresses this cost by learning to infer representations across a collection of training inputs, rather than fitting each new input separately. TransINR (Chen & Wang, 2022) and GINR (Kim et al., 2023) learn to predict or modulate the parameters of an implicit neural representation, while FastNeRV (Chen et al., 2024) applies weight prediction to video encoding. TeCoNeRV (Padmanabhan et al., 2026) extends this approach to patch tubelets and encourages temporally coherent weights for residual storage. At test time, video-specific information is inferred rather than fitted, but remains expressed through decoder parameters or their modulation.

A complementary approach stores input-specific information in latent codes, or tokens, read by a decoder with weights shared across inputs. Attention-based neural fields (Zhang et al., 2024) use this formulation, and CoANeRV (Guo et al., 2026) applies it to feed-forward video representation. Instead of predicting decoder weights, an encoder forms a video token bank. A coordinate query identifies an output location and time and retrieves relevant content through attention (Fig. 1(b)). In CoANeRV, queries from different frames and regions access the same global memory. This memory can learn temporal persistence and spatial relevance, but its layout does not explicitly separate shared from frame-specific information or assign codes to output regions.

Under a fixed storage budget, video structure motivates both temporal sharing and spatial ownership. In the juggling sequence in Fig. 1(a), much of the background persists while changes concentrate around the hands and ball. Temporal decomposition is already explored by the static and dynamic

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.



(a) Temporal Observation (b) Video State And Readout (c) Selected-Token Zeroing

Figure 1: **Motivation and representation. (a)** Consecutive frames share content but exhibit local changes; green/yellow boxes mark smaller/larger frame differences. **(b)** Representation and readout: FastNeRV (Chen et al., 2024) predicts decoder weights; CoANeRV (Guo et al., 2026) attends to global tokens; PAIRNeRV reads patch-owned appearance and innovation codes. Purple outlines connect each queried patch to its codes. **(c)** Post-encoding interventions, relative to the unedited reconstruction: zeroing appearance changes its patch across the shown frames, whereas zeroing innovation at T_0 changes only that frame's patch. These edits reveal distinct temporal scopes.

codes of DS-NeRV (Yan et al., 2024), and CoordTok (Jang et al., 2025) uses factorized feature planes for coordinate-based patch reconstruction. Our focus is on coupling these temporal roles with explicit spatial ownership in a feed-forward representation under a fixed storage ceiling. Gathering context across patches during encoding need not require every reconstruction query to revisit the full video memory. Shared codes are stored once, while frame-specific codes repeat across frames; both must fit the same bit budget.

We propose **PAIRNeRV**, a feed-forward neural video representation that organizes state by temporal role and spatial ownership (Fig. 2). A clip-wide encoder gathers contextualized frame–patch features. Two code writers then produce a patch-persistent appearance code shared across frames and a frame-local innovation code for each frame–patch pair. Both are assigned to spatial patches, defining where they are read without restricting where their information originates. The codes are learned from reconstruction without separate appearance or motion targets. A shared multilayer perceptron reconstructs requested patches from their codes and spatial coordinates, without reading the full video state. Time-invariant appearance and geometry projections are reused across frames.

Storage precision complements this organization. Motivated by CoANeRV's small reconstruction losses with eight-bit storage (Guo et al., 2026), we allocate more code values to the two temporal roles within the same bit budget. Only stored codes are quantized; training and rendering remain floating point. Our contributions are summarized as follows:

- We introduce **PAIRNeRV**, a feed-forward video representation that couples temporal roles with explicit spatial ownership. Clip-shared appearance codes and frame-specific innovation codes are assigned to output patches, organizing persistent and changing content within a fixed per-clip storage budget.
- We design a **shared patch-local renderer** that reconstructs pixels through direct lookup of the corresponding appearance and innovation codes, without global retrieval attention. This design supports selective patch reconstruction and reuses appearance and geometry projections across frames.
- Experiments demonstrate improved reconstruction quality under a common video-state budget, transfer to unseen datasets without fine-tuning, and faster resident-state decoding than a global-attention baseline. Controlled studies examine the contributions of temporal roles, local readout, storage allocation, and cross-frame reuse.

2 Related Work

Neural Video Representations. NeRV (Chen et al., 2021a) represents a video with a neural function mapping temporal indices to frames. Subsequent methods improve reconstruction through spatial–temporal context, learned embeddings, and hierarchical encodings (Li et al., 2022; Chen et al.,

2023; Kwan et al., 2023). Other designs exploit temporal and frequency structure (Zhao et al., 2023; Kim et al., 2024b; Saethre et al., 2024; Shin et al., 2024), pyramids (Zhao et al., 2024), or feature trees (Zhao et al., 2025). DS-NeRV (Yan et al., 2024) separates static and dynamic codes, making temporal sharing an explicit representation prior. NIRVANA (Maiya et al., 2023) fits patch-wise predictors with cross-group weight initialization. Unlike these fitted representations, PAIRNeRV infers temporally structured, patch-owned codes without per-video optimization.

Amortized Representation Learning. TransINR (Chen & Wang, 2022) predicts instance-specific weights with a Transformer, whereas GINR (Kim et al., 2023) modulates a subset of otherwise shared weights. For videos, MetaNeRV (Guo et al., 2025) learns an initialization for rapid adaptation, and FastNeRV (Chen et al., 2024) predicts NeRV parameters in one forward pass. TeCoNeRV (Padmanabhan et al., 2026) predicts patch-tubelet parameters that modulate shared base weights, exploiting inter-clip similarity through temporal regularization and residual parameter storage. Its spatial decomposition applies to predicted decoder parameters. Partitioning also accelerates INR fitting (Liu et al., 2023) and supports weight-space cropping (Ashkenazi & Treister, 2024). PAIRNeRV instead assigns video codes to patches while sharing decoder weights.

Token-Conditioned Neural Fields. Token-conditioned fields store instance content in codes read by a shared decoder. LIIF (Chen et al., 2021b) queries local image features; LAG-INR (Lee et al., 2023) aggregates tokens according to the query coordinate, while ANR (Zhang et al., 2024) combines coordinate and instance features through localized attention. Tokenized Neural Fields (Haider & Rosenbaum, 2025) fits structured tokens; LH-NeF (Urbano et al., 2026) combines local token groups with query-dependent routing and cross-attention. CoordTok (Jang et al., 2025) queries spatial and spatiotemporal feature planes, then fuses features through self-attention among reconstruction patches. CoANeRV (Guo et al., 2026) reads a clip-wide token bank through coordinate-conditioned cross-attention. Although its queries are grouped into spatial blocks, each block still accesses the full bank. PAIRNeRV pairs clip-shared appearance with frame-indexed innovation. Each query reads its patch's codes directly, without bank-wide attention or output-patch fusion; global context enters during code formation.

Video Compression. Conventional codecs combine prediction, transforms, and entropy coding (Wiegand et al., 2003; Sullivan et al., 2012); learned codecs develop conditional representations and efficient execution (Li et al., 2024; Tang et al., 2025; Jia et al., 2025). HiNeRV (Kwan et al., 2023), C3 (Kim et al., 2024a), and NVRC (Kwan et al., 2024) compress neural representations through quantization and coding. PAIRNeRV instead focuses on storage precision and code allocation under a fixed state budget per clip. This budget differs from the entropy-coded bitrate of a video codec.

3 METHOD

Building on CoANeRV's token-space formulation (Guo et al., 2026), PAIRNeRV encodes video as patch-owned codes and reconstructs pixels with a shared coordinate decoder. The codes form video state. Figure 2 shows global encoding, token writing, fixed-bit storage, and patch-local rendering.

Let $V \in \mathbb{R}^{T\times C\times H\times W}$ denote a clip of T frames, C channels, and resolution $H \times W$. The encoder and writers E_ϕ infer continuous state $\mathcal{Z}_V$; decoder D_ψ predicts pixel values at $q = (t, x, y)$:

$$\mathcal{Z}_V = E_\phi(V), \qquad \hat{V}(q) = D_\psi(\mathrm{Read}(\mathcal{Z}_V; q), q)\,. \tag{1}$$

Here $t \in \{1, \dots, T\}$ indexes frames, $(x, y) \in [0, 1]^2$ gives normalized coordinates, and Read retrieves the query's codes. Only $\mathcal{Z}_V$ is video-specific; parameters ϕ, ψ are shared.

3.1 GLOBAL ENCODING

We gather clip-wide context before writing patch codes, while keeping reconstruction patch-local. A shared convolutional tokenizer maps each frame to $P = (H/R)(W/R)$ non-overlapping patches of side length R. Each feature receives a learned spatial embedding and a frame-time embedding.

A residual Transformer (Vaswani et al., 2017) processes the full frame–patch sequence. We add the original, unembedded tokenizer features to the corresponding outputs and restore the frame–patch layout to obtain $\mathbf{H} \in \mathbb{R}^{T\times P\times d}$, where d is the feature width. Each $\mathbf{H}_{t,p}$, with $p \in \{1, \dots, P\}$, retains its frame–patch indices while incorporating clip-wide context. The writers can therefore assign video-wide evidence to individual patches (Appendix B.1).

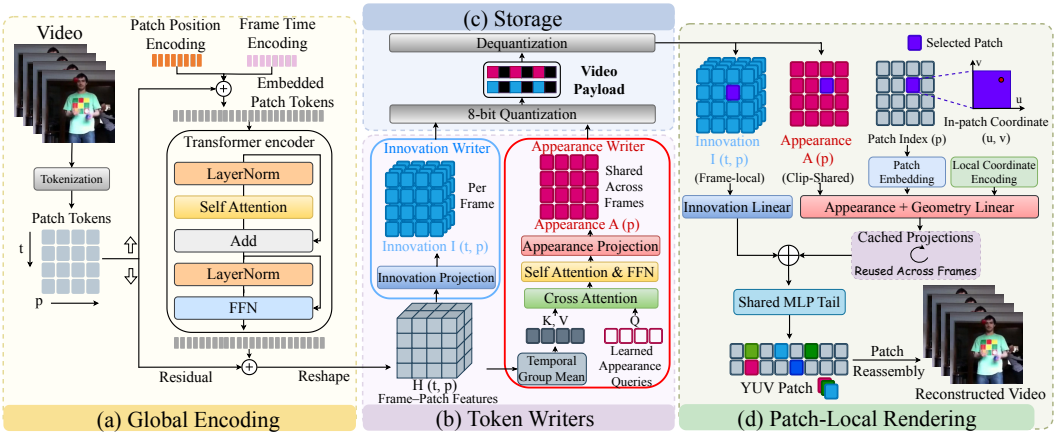


Figure 2: **PAIRNeRV framework.** (a) A Transformer encoder gathers clip-wide context; its outer residual retains the original patch features. (b) Grouped temporal pooling and learned queries form one clip-shared appearance code per patch; direct projection forms one innovation code per frame–patch pair. (c) Quantized codes and their metadata form the video payload. (d) A shared renderer combines the retrieved codes with geometry, reusing appearance and geometry projections across frames while evaluating the nonlinear tail per pixel. Thus each output uses clip-wide encoded context through codes assigned to its own patch.

3.2 Token Writers

Temporal persistence calls for shared capacity, while changes require frame-specific state. Two writers produce a *patch-persistent appearance code* $\mathbf{A}_p \in \mathbb{R}^{d_a}$, stored once per patch, and a *frame-local innovation code* $\mathbf{I}_{t,p} \in \mathbb{R}^{d_i}$, stored for each frame–patch pair. The widths d_a, d_i count stored values per code. The roles govern storage and access, not a prescribed semantic decomposition.

Appearance writer. To form a code shared across frames, we first average $\mathbf{H}$ within G consecutive frame groups at each patch, giving memory $\overline{\mathbf{H}} \in \mathbb{R}^{G\times P\times d}$. Pooling reduces the temporal extent of this memory; retaining multiple groups preserves coarse temporal structure rather than collapsing the entire clip to one mean. Following the learned-query token formation in CoANeRV (Guo et al., 2026), P learned appearance queries $\mathbf{Q}^A \in \mathbb{R}^{P\times d}$ cross-attend to this memory. The queries are shared parameters; pooled features supply video-dependent keys and values. Self-attention and a feed-forward network refine the outputs before projection to width d_a. Denoting this complete readout by F_A, we obtain

$$\mathbf{A} = F_A(\mathbf{Q}^A, \overline{\mathbf{H}}), \qquad \mathbf{A} \in \mathbb{R}^{P\times d_a}. \tag{2}$$

Each output slot p supplies one clip-shared appearance code for patch p, using all GP memory entries. The writer thus combines global context with patch-specific ownership at readout.

Innovation writer. The second writer retains the frame–patch layout so that each frame has its own local state. A shared affine map F_I projects the contextualized features without pooling or further attention:

$$\mathbf{I}_{t,p} = F_I(\mathbf{H}_{t,p}), \qquad \mathbf{I} \in \mathbb{R}^{T\times P\times d_i}. \tag{3}$$

This branch receives global context through $\mathbf{H}_{t,p}$ while preserving a separate output for each frame–patch pair. Innovation is learned directly from reconstruction rather than computed as a residual. Together, the writers produce the continuous video state

$$\mathcal{Z}_V = (\mathbf{A}, \mathbf{I}). \tag{4}$$

3.3 Fixed-Bit Storage

Storage precision and code width share a fixed budget. Following CoANeRV's observation of small losses from eight-bit storage (Guo et al., 2026), we use signed eight-bit integers (Q8) to retain more code values under a fixed bit ceiling. Each video's code families are symmetrically quantized with separate scales. The integer codes, scales, and allocation identifier form payload $\mathcal{S}_V$, excluding encoder features and shared weights. The storage round trip is

$$\mathcal{Z}_V \xrightarrow{\text{quantize}} \mathcal{S}_V \xrightarrow{\text{dequantize}} \widehat{\mathcal{Z}}_V = (\widehat{\mathbf{A}}, \widehat{\mathbf{I}}). \tag{5}$$

Here the hat denotes recovered floating-point codes, which replace the continuous state in Eq. 1 for stored-state rendering. Quantization is used for storage and evaluation, not training; the shared renderer still operates in floating point.

More generally, with b_a and b_i bits per appearance and innovation value, the payload must satisfy

$$B(\mathcal{S}_V) = Pb_a d_a + TPb_i d_i + B_{\text{meta}} \leq B_{\max}, \tag{6}$$

where B_{meta} counts metadata bits and $B_{\max}$ is the budget. Innovation storage grows with clip length, whereas appearance is stored once per patch. We therefore select their widths jointly under this budget, without assuming entropy-coding gains (Appendix D.1).

3.4 Patch-Local Rendering

Explicit ownership replaces attention over the full state with local lookup. For $q = (t, x, y)$, the spatial partition identifies patch $p = \pi(x, y)$ and normalized local coordinate $\boldsymbol{\xi} = (u, v) \in [0, 1]^2$. The query retrieves the appearance code of that patch and its innovation code at the requested frame:

$$\text{Read}(\widehat{\mathcal{Z}}_V; q) = \left(\widehat{\mathbf{A}}_{\pi(x,y)}, \widehat{\mathbf{I}}_{t,\pi(x,y)}\right). \tag{7}$$

Thus, each query accesses $d_a + d_i$ code values rather than attending to the full video state. The frame index selects a stored innovation code; it is not a continuous-time input to the rendering network. With other recovered codes and shared weights fixed, appearance edits affect only their patch across frames; innovation edits affect only their frame–patch pair. This locality concerns stored-state readout, not input edits before global encoding.

Shared reconstruction. The codes describe a patch; geometry locates its pixels. Shared affine maps f_a, f_i lift the recovered codes to features $\widetilde{\mathbf{A}}_p = f_a(\widehat{\mathbf{A}}_p)$ and $\widetilde{\mathbf{I}}_{t,p} = f_i(\widehat{\mathbf{I}}_{t,p})$. Tildes denote decoder features, not additional stored state. A learned embedding $\mathbf{e}_p$ identifies the patch, while $\ell(\boldsymbol{\xi})$ concatenates local coordinates and their Fourier features (Tancik et al., 2020). These geometric inputs let pixels within a patch share the same codes while producing different output values.

A shared multilayer perceptron (MLP) combines the code and geometry features. We separate its first affine layer from the remaining nonlinear layers g_ψ to make the reusable computation explicit:

$$\mathbf{h}_{t,p,\boldsymbol{\xi}} = \underbrace{W_a\widetilde{\mathbf{A}}_p + W_p\mathbf{e}_p + W_\xi\ell(\boldsymbol{\xi}) + \mathbf{b}}_{\text{reused across frames}} + \underbrace{W_i\widetilde{\mathbf{I}}_{t,p}}_{\text{frame-specific}}, \tag{8}$$

$$\hat{V}(q) = g_\psi(\mathbf{h}_{t,p,\boldsymbol{\xi}}). \tag{9}$$

Here W_a, W_i, W_p, W_ξ map inputs to the same hidden width, and $\mathbf{b}$ is a shared bias. In Fig. 2(d), *Innovation Linear* groups f_i with W_i, while *Appearance + Geometry Linear* groups f_a with the remaining first-layer terms. Their sum feeds the *Shared MLP Tail*, g_ψ. Reassembling the rendered patch grids yields a frame without using earlier reconstructions.

Cross-frame reuse. Equation 8 separates terms by their temporal scope. The appearance projection is computed once per patch in a clip; the geometry projections depend only on the shared model and coordinate grid. We cache these projections separately. For each requested frame, we then add the innovation contribution, computed once per frame–patch pair and shared by pixels within that patch. Reuse stops at the first-layer preactivation; the nonlinear tail still runs per pixel. This changes evaluation order but preserves the function in exact arithmetic (Appendix B.2).

Training and inference. We jointly train the encoder, both writers, and renderer on continuous state using the reconstruction loss

$$\mathcal{L}_{\text{rec}} = \frac{\|V - \hat{V}\|_2^2}{CTHW}. \tag{10}$$

No separate appearance or motion targets are required. Stored-state evaluation uses the quantization round trip in Eq. 5. For an unseen clip, encoding and writing run once, without per-video optimization or weight prediction. Subsequent queries use only recovered state and the shared renderer, batched over patches or frames. Long videos use successive clips.

4 Experiments

We test reconstruction at fixed storage, examine temporal roles and patch-local readout, and measure precision, capacity, and rendering cost.

4.1 Evaluation Protocol

Protocol. Following CoANeRV (Guo et al., 2026), we train each clip-length setting on its fixed 10,000-video Kinetics-400 (K400) subset (Kay et al., 2017). The trained model encodes each clip from Train (that subset), K400 validation, Something-Something V2 (SthV2) (Goyal et al., 2017), and UCF101 (Soomro et al., 2012) in one forward pass, without target-dataset adaptation. Clips contain the first $F \in \{4, 8, 16\}$ frames at 256×256 in normalized luminance–chrominance (YUV) values. We report PSNR and structural similarity (SSIM) (Wang et al., 2004) at 150 epochs and after 20 epochs; implementation details are in Appendix A. Allocations are selected on K400 validation and fixed for cross-dataset evaluation (Appendix A.4).

Reconstruction comparisons. PAIRNeRV stores 872.5 kbit per clip within CoANeRV's 884.7-kbit state ceiling and reconstructs from recovered Q8 codes. The budget excludes shared weights and entropy coding. Baseline state is counted in FP32; FP32/Q8 denotes storage, not computation precision (Appendix A.2).

Model size and cost. Encoder size includes both writers; INR size counts the shared reconstruction path. Their widths vary with clip length, so INR size need not grow monotonically. Training-step and rendering comparisons use matched execution conditions; reconstruction quality is reported separately.

Table 1: **Reconstruction comparison.** F is clip length; INR size counts reconstruction parameters. Baselines are reported by CoANeRV (Guo et al., 2026); PAIRNeRV is evaluated here, with estimated GPU-hours. Bold/underline mark the best/second-best reported quality.

Method	F	Encoder size	INR size	Video state (kbit) ↓	Epoch	GPU hrs ↓	PSNR ↑				SSIM ↑			
							Train	K400	SthV2	UCF101	Train	K400	SthV2	UCF101
TransINR	4	48.0M	99.0k	798.7	150	63	23.7	22.1	24.6	22.1	.659	.631	.728	.622
GINR	4	47.6M	139.4k	819.2	150	65	24.5	23.2	25.9	23.1	.685	.660	.744	.660
FastNeRV	4	47.6M	85.6k	771.2	150	9	26.6	26.6	29.4	26.1	.756	.754	.816	.752
CoANeRV	4	45.5M	86.4k	884.7	150	39	35.3	34.8	36.0	35.8	.955	.951	.956	.955
PAIRNeRV	4	44.9M	54.9k	872.5	20	1	35.0	35.0	37.8	36.4	.951	.949	.960	.956
PAIRNeRV	4	44.9M	54.9k	872.5	150	11	**39.8**	**39.5**	**42.6**	**42.1**	**.978**	**.976**	**.982**	**.982**
TransINR	8	48.0M	99.0k	798.7	150	119	22.3	20.3	22.8	20.7	.626	.595	.703	.591
GINR	8	47.6M	139.4k	819.2	150	123	23.9	22.8	25.3	22.7	.671	.650	.737	.651
FastNeRV	8	47.6M	85.6k	771.2	150	11	25.8	25.8	28.5	25.2	.732	.727	.795	.723
CoANeRV	8	46.3M	86.4k	884.7	150	70	33.1	32.1	32.8	33.3	.939	.927	.935	.938
PAIRNeRV	8	44.9M	75.1k	872.5	20	2	34.3	34.3	36.9	35.6	.946	.944	.955	.953
PAIRNeRV	8	44.9M	75.1k	872.5	150	15	**37.3**	**37.2**	**39.6**	**39.3**	**.968**	**.966**	**.971**	**.974**
TransINR	16	48.0M	99.0k	798.7	150	234	21.5	18.4	21.1	19.2	.615	.555	.678	.561
GINR	16	47.6M	139.4k	819.2	150	242	22.9	21.7	24.2	21.7	.647	.624	.720	.625
FastNeRV	16	47.6M	85.6k	771.2	150	15	23.6	23.2	25.9	22.9	.657	.642	.731	.642
CoANeRV	16	47.9M	86.4k	884.7	150	128	31.5	30.9	31.1	31.3	.923	.917	.918	.919
PAIRNeRV	16	44.9M	72.1k	872.5	20	4	33.2	33.2	35.6	34.3	.936	.934	.944	.943
PAIRNeRV	16	44.9M	72.1k	872.5	150	29	**34.5**	**34.4**	**36.3**	**36.0**	**.948**	**.946**	**.950**	**.956**

4.2 Reconstruction and Transfer

Reconstruction quality. PAIRNeRV improves reconstruction within the common video-state ceiling (Table 1). At E150, its K400 PSNR exceeds published CoANeRV results by 4.7, 5.1, and 3.5 dB for F4/F8/F16, with higher SSIM in each setting. These gains do not require a larger decoder: PAIRNeRV uses 54.9–75.1k INR parameters, compared with CoANeRV's 86.4k. Quality decreases as the configurations represent more frames within the same per-clip storage ceiling, but PAIRNeRV retains an advantage at all three lengths.

Cross-dataset inference. PAIRNeRV retains its advantage on SthV2 and UCF101. At F8, it reaches 39.6 dB on SthV2 and 39.3 dB on UCF101, compared with CoANeRV's 32.8 and 33.3 dB. The same PSNR and SSIM ordering holds at F4 and F16.

Early quality. At E20, PAIRNeRV already exceeds CoANeRV's E150 K400 PSNR at each clip length; its own reconstruction improves further by E150. Matched training-step costs appear in Tables 2 and 3.

Qualitative comparison. Figure 3 shows how the quality gains appear in local image structure. PAIRNeRV retains finger boundaries, printed digits, and the drum-rim contour more faithfully than the compared reconstructions, which show stronger smoothing or structured artifacts. These crops complement the dataset-level PSNR and SSIM results.

Native-resolution detail. Figure 4 extends the comparison to UVG and MCL frames without shrinking the full image to 256×256. Both methods reconstruct the same frames from independently

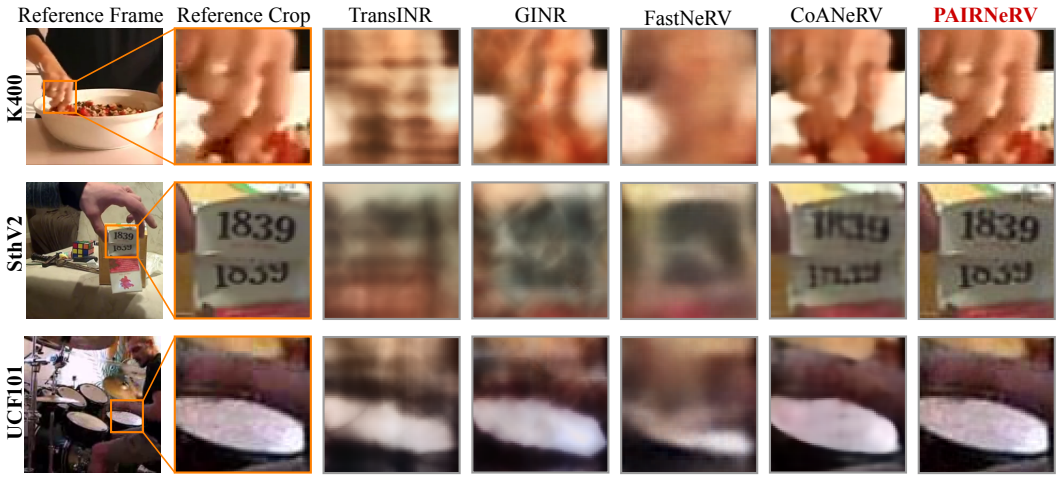


Figure 3: **Qualitative comparison.** Orange boxes mark corresponding enlarged regions in each row. PAIRNeRV retains clearer object boundaries and text in the displayed crops.

encoded 256×256 tiles. PAIRNeRV retains more of the shown moving contours and fabric texture than CoANeRV. The selected crops illustrate these differences rather than a dataset-wide quality estimate; matched-window scores appear in Appendix Table 10.

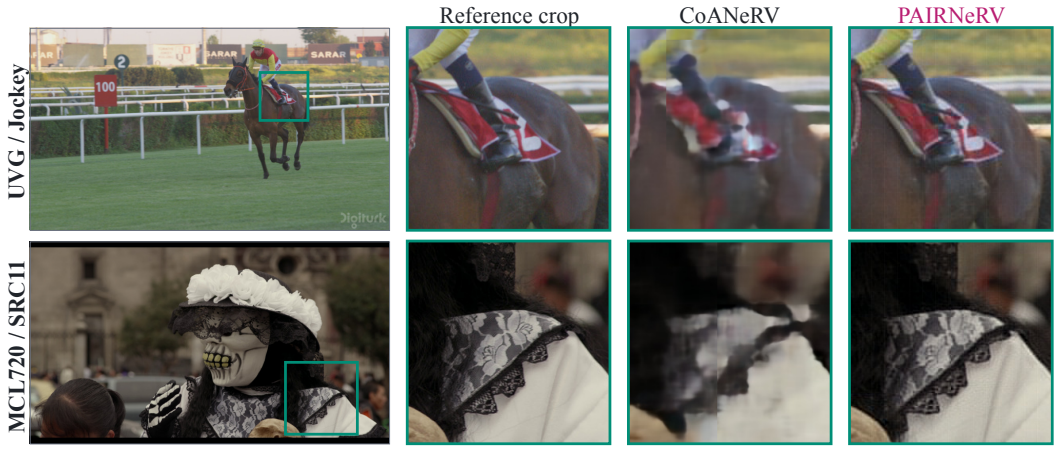


Figure 4: **Native-resolution qualitative comparison.** Boxes locate the same 256×256 regions enlarged at right in UVG Jockey and MCL720 SRC11. Compared with the feed-forward CoANeRV baseline, PAIRNeRV better preserves the saddle contour and patterned costume in these examples.

4.3 Roles and Ownership

Role-preserving readout. Table 2 examines how temporal roles and spatial ownership affect readout. Among global readers, accessing A and I separately (d), rather than reading their joint bank (c), improves PSNR and SSIM on all four splits. Both readers return two vectors under the same stored budget, so the gain does not come from adding a second output. It supports preserving the distinction between the two roles when retrieving their content. On K400, the seed-paired PSNR gain is 0.36 ± 0.18 dB (Appendix Table 6).

Patch-local lookup. Replacing separate global retrieval (d) with direct lookup (e) improves PSNR further while reducing execution cost. The paired K400 gain is 0.24 ± 0.05 dB. It reduces the training step from 30.92 to 27.56 ms and lowers cold decoding latency from 3.21 to 2.62 ms. Thus, assigning each output patch its own codes is useful for both reconstruction and rendering efficiency. Local readout also improves All-I reconstruction across the four splits (a versus b), showing that locality remains beneficial even without shared appearance state.

Shared appearance versus All-I. The A/I split improves Train, K400, and UCF101 PSNR over local All-I (b), but All-I retains higher SthV2 PSNR and a shorter training step. All-I assigns its stored budget entirely to frame-specific codes; the full model (e) instead reserves part for appearance

Table 2: **Roles and ownership ablation.** A/I denotes separate appearance and innovation state; separate read restricts each output to its corresponding bank. Cold decode includes cache construction, and "–" denotes an inapplicable setting. Row (e) is PAIRNeRV: local readout improves quality and cold decoding over separate global readout.

	Stored state	Separate A/I read	Local read	Payload (kbit)	Train step (ms/step) ↓	Cold decode (ms/clip) ↓		PSNR (dB) ↑				SSIM ↑			
						Reuse off	Reuse on	Train	K400	SthV2	UCF101	Train	K400	SthV2	UCF101
(a)	All-I	–	×	868.4	27.77	3.27	–	35.6	35.5	38.2	37.0	0.952	0.950	0.960	0.957
(b)	All-I	–	✓	868.4	25.32	2.59	2.63	36.6	36.4	39.2	38.1	0.960	0.958	0.966	0.964
(c)	A/I	×	×	872.5	30.26	3.24	–	36.4	36.3	38.5	38.2	0.958	0.956	0.962	0.964
(d)	A/I	✓	×	872.5	30.92	3.21	–	36.8	36.6	38.9	38.6	0.962	0.960	0.965	0.967
(e)	**A/I**	✓	✓	872.5	27.56	2.62	2.65	37.0	36.9	39.1	39.0	0.964	0.962	0.966	0.970

stored once per clip and reused across frames. This comparison reveals a capacity trade-off, not a uniform advantage from adding appearance to every setting. Section 4.4 examines how this balance changes with clip length and dataset.

Spatial and temporal scope. Figure 1(c) illustrates readout locality after encoding: zeroing a patch's appearance affects its reconstruction across frames, whereas zeroing its innovation at T_0 affects only that frame's patch. The intervention links the assigned code roles to their spatial and temporal readout scope: globally encoded information can still be accessed locally. It does not imply semantic disentanglement of appearance and motion.

4.4 Precision and Capacity

Storage precision. Q8 reduces PAIRNeRV's payload by about 75% while lowering PSNR by only 0.01–0.24 dB for the same learned state (Table 3). The saved bits can support more code values within a fixed storage budget.

Table 3: **Precision and capacity.** FP32 and Q8 denote stored-state precision; each pair shares one trained model. Underlining marks the common state-size ceiling, and bold marks the better quality or lower step time within a comparison. Expanded Q8 outperforms compact FP32 at equal storage.

F	Method	Stored values (k)	Payload (kbit)		Train step (ms/step) ↓	Train		K400		SthV2		UCF101	
			FP32	Q8		FP32	Q8	FP32	Q8	FP32	Q8	FP32	Q8
Compact states													
4	CoANeRV, N=384	27.6	884.7	221.2	76.59	**34.93**	**34.89**	32.77	32.75	35.38	35.35	34.05	34.03
	PAIRNeRV	27.6	884.7	221.2	**22.39**	34.41	34.37	**34.30**	**34.26**	**36.82**	**36.77**	**35.76**	**35.72**
8	CoANeRV, N=384	27.6	884.7	221.2	86.98	**32.24**	**32.22**	29.57	29.56	31.25	31.24	30.73	30.72
	PAIRNeRV	27.6	884.7	221.2	**27.91**	32.23	32.21	**32.16**	**32.14**	**34.24**	**34.22**	**33.32**	**33.30**
16	CoANeRV, N=384	27.6	884.7	221.2	131.97	30.12	30.11	26.90	26.90	27.88	27.88	27.96	27.95
	PAIRNeRV	27.6	884.7	221.2	**44.68**	**30.32**	**30.31**	**30.28**	**30.27**	**32.30**	**32.29**	**31.08**	**31.07**
Expanded states													
4	CoANeRV, N=1535	110.5	3536.6	884.2	90.49	38.88	38.73	37.55	37.44	40.02	39.86	39.32	39.18
	PAIRNeRV	109.1	3489.8	872.5	**22.02**	**39.67**	**39.50**	**39.46**	**39.30**	**42.37**	**42.14**	**41.97**	**41.74**
8	CoANeRV, N=1535	110.5	3536.6	884.2	131.77	36.13	36.08	34.72	34.68	36.70	36.65	36.37	36.32
	PAIRNeRV	109.1	3489.8	872.5	**27.98**	**37.15**	**37.05**	**36.99**	**36.89**	**39.18**	**39.05**	**39.10**	**38.97**
16	CoANeRV, N=1535	110.5	3536.6	884.2	234.59	31.28	31.26	28.61	28.61	29.62	29.62	29.78	29.78
	PAIRNeRV	109.1	3489.8	872.5	**45.07**	**34.45**	**34.40**	**34.34**	**34.29**	**36.25**	**36.18**	**35.93**	**35.87**

Capacity at a fixed budget. More code values improve quality at the same bit budget: independently trained expanded Q8 states store 3.94× more values and gain 3.88–5.98 dB over compact FP32 at a slightly smaller payload (Table 3).

Comparison with token expansion. Expanded Q8 PAIRNeRV exceeds expanded CoANeRV on all four splits at each clip length, using fewer bits (Table 3). At F8, K400 PSNR is 36.89 versus 34.68 dB; expansion increases training-step time by 51% for CoANeRV but less than 1% for PAIRNeRV. Three-seed F8 results retain the advantage (Appendix Table 7).

Appearance–innovation balance. The preferred split varies with clip length and dataset (Fig. 5). One allocation leads all four datasets at F4. Excessive appearance storage hurts F8/F16 quality, while the smallest F16 appearance share also hurts three splits: neither role uniformly benefits from more capacity.

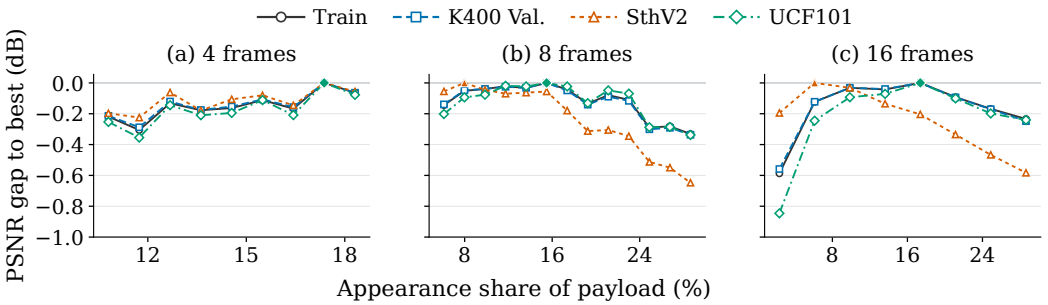


Figure 5: **Appearance–innovation allocation.** Each curve reports PSNR relative to its best measured allocation under the same storage budget; filled markers denote those maxima. The preferred appearance share varies with clip length and dataset.

4.5 Rendering Efficiency

Dense decoding, cache reuse, and selective access use distinct timing boundaries (Fig. 6).

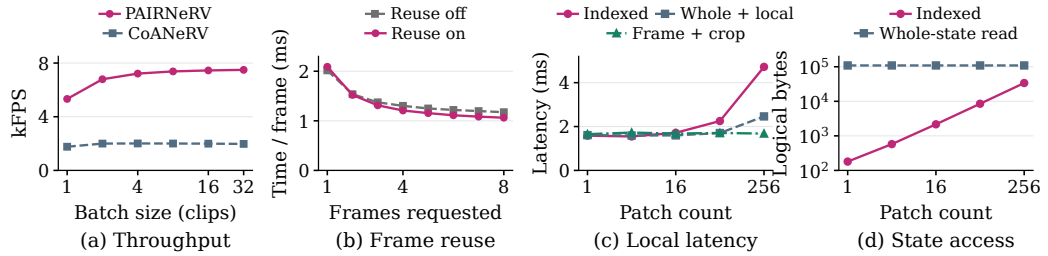


Figure 6: **Rendering and access efficiency. (a)** Resident-state throughput; **(b)** amortized latency including initial cache construction; **(c,d)** file-to-region latency and logical bytes read. Shading indicates variation across requests. Local rendering improves throughput, while reduced reads yield request-dependent latency gains.

Dense decoding. PAIRNeRV delivers 3.0–3.8× higher warm resident-state throughput than native CoANeRV across the tested batches (Fig. 6(a)). The first decode of a new resident clip is 1.95× faster at batch one. These are decoder, not end-to-end codec, measurements.

Cross-frame reuse. Appearance and geometry projection reuse lowers eight-request amortized latency from 1.17 to 1.06 ms per frame, including cache construction (Fig. 6(b)). Cold full-clip latency barely changes because projections are already shared (Table 2).

Selective access. For one patch, indexed access reduces logical reads from 109,104 to 180 bytes (Fig. 6(c,d)). Median latency remains similar to whole-state local rendering (1.58 versus 1.61 ms, with overlapping interquartile ranges). Indexed access is faster than full-frame rendering for one- and four-patch requests, but not for larger requests. All access paths reconstruct the same requested pixels within numerical tolerance (Appendix E.3), so selective access changes the execution scope rather than the reconstruction target.

High-resolution region requests. At 1080p, rendering the requested tiles takes 1.68 ms for one 64×64 region, versus 22.24 ms for a full-frame render (Appendix Table 9). For dispersed regions, batched tiles outperform patch indexing.

5 Conclusion and Limitations

PAIRNeRV pairs patch-owned, clip-shared appearance with frame-specific innovation under a fixed state budget. Local readout supports selective reconstruction; cross-frame reuse reduces repeated projection costs. The design separates global context gathering from local reconstruction, allowing each query to use contextualized codes without revisiting the full video state. Experiments show improved reconstruction and cross-dataset transfer, with faster resident-state decoding than the tested global-attention baseline. Selective-access gains remain request-dependent. Independent high-resolution tiles and weaker rate–distortion performance than conventional codecs remain limitations; joint spatiotemporal modeling and coding offer directions for future work.

AI USE STATEMENT

AI tools assisted with language editing, implementation, and result analysis, and provided feedback on experimental design. The authors reviewed the AI-assisted outputs and take responsibility for the final manuscript.

ETHICS STATEMENT

This work studies video representation and reconstruction using established research datasets. It introduces no new human-subject data collection and does not perform identity inference or other decisions about individuals. Dataset use should follow the licenses and access conditions of the original sources. As reconstruction systems can reproduce visual content present in their input, deployment should respect the source videos' privacy, copyright, and consent requirements.

REPRODUCIBILITY STATEMENT

The method definition, fixed-bit accounting, optimization objective, and main experimental protocol are described in the main paper. The appendix provides code-family dimensions, quantization rules, preprocessing, training schedules, checkpoint selection, runtime measurement, and additional ablations. Published baseline results are distinguished from locally trained comparisons, and the supplement specifies the evaluation settings for each.

REFERENCES

Maor Ashkenazi and Eran Treister. Towards croppable implicit neural representations. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

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*, volume 34, pp. 21557–21568, 2021a.

Hao Chen, Matthew Gwilliam, Ser-Nam Lim, and Abhinav Shrivastava. HNeRV: A hybrid neural representation for videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10270–10279, 2023.

Hao Chen, Saining Xie, Ser-Nam Lim, and Abhinav Shrivastava. Fast encoding and decoding for implicit video representation. In *European Conference on Computer Vision*, pp. 402–418, 2024.

Yinbo Chen and Xiaolong Wang. Transformers as meta-learners for implicit neural representations. In *European Conference on Computer Vision*, pp. 170–187, 2022.

Yinbo Chen, Sifei Liu, and Xiaolong Wang. Learning continuous image representation with local implicit image function. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8628–8638, 2021b.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018.

Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, Florian Hoppe, Christian Thurau, Ingo Bax, and Roland Memisevic. The "something something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5842–5850, 2017.

Jialong Guo, Ke Liu, Jiangchao Yao, Zhihua Wang, Jiajun Bu, and Haishuai Wang. MetaNeRV: Meta neural representations for videos with spatial-temporal guidance. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 3257–3265, 2025.

Jialong Guo, Ke Liu, Mengxuan Li, Jiajun Bu, and Haishuai Wang. CoANeRV: Coordinate-aware token-space neural video representation. *arXiv preprint arXiv:2608.13938*, 2026.

Azmi Haider and Dan Rosenbaum. Tokenized neural fields: Structured representations of continuous signals. In *NeurIPS 2025 Workshop on Structured Probabilistic Inference and Generative Modeling*, 2025.

Jingning Han, Bohan Li, Debargha Mukherjee, Ching-Han Chiang, Adrian Grange, Cheng Chen, Hui Su, Sarah Parker, Sai Deng, Urvang Joshi, Yue Chen, Yunqing Wang, Paul Wilkins, Yaowu Xu, and James Bankoski. A technical overview of AV1. *Proceedings of the IEEE*, 109(9):1435–1462, 2021.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

ITU-R. Parameter values for the HDTV standards for production and international programme exchange. Recommendation BT.709-6, International Telecommunication Union, 2015.

Huiwon Jang, Sihyun Yu, Jinwoo Shin, Pieter Abbeel, and Younggyo Seo. Efficient long video tokenization via coordinate-based patch reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22853–22863, 2025.

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.

Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

Chiheon Kim, Doyup Lee, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Generalizable implicit neural representations via instance pattern composers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11808–11817, 2023.

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, 2024a.

Jina Kim, Jihoo Lee, and Je-Won Kang. SNeRV: Spectra-preserving neural representation for video. In *European Conference on Computer Vision*, pp. 332–348, 2024b.

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*, volume 36, 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*, volume 37, 2024.

Doyup Lee, Chiheon Kim, Minsu Cho, and Wook-Shin Han. Locality-aware generalizable implicit neural representation. In *Advances in Neural Information Processing Systems*, volume 36, 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.

Zizhang Li, Mengmeng Wang, Huaijin Pi, Kechun Xu, Jianbiao Mei, and Yong Liu. E-NeRV: Expedite neural video representation with disentangled spatial-temporal context. In *European Conference on Computer Vision*, pp. 267–284, 2022.

Ke Liu, Feng Liu, Haishuai Wang, Ning Ma, Jiajun Bu, and Bo Han. Partition speeds up learning implicit neural representations based on exponential-increase hypothesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5474–5483, 2023.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Shishira R. Maiya, Sharath Girish, Max Ehrlich, Hanyu Wang, Kwot Sin Lee, Patrick Poirson, Pengxiang Wu, Chen Wang, and Abhinav Shrivastava. NIRVANA: Neural implicit representations of videos with adaptive networks and autoregressive patch-wise modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14378–14387, 2023.

Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Practical full resolution learned lossless image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10629–10638, 2019.

Alexandre Mercat, Marko Viitanen, and Jarno Vanne. UVG dataset: 50/120fps 4k sequences for video codec analysis and development. In *Proceedings of the ACM Multimedia Systems Conference*, pp. 297–302, 2020.

Namitha Padmanabhan, Matthew Gwilliam, and Abhinav Shrivastava. TeCoNeRV: Leveraging temporal coherence for compressible neural representations for videos. *arXiv preprint arXiv:2602.16711*, 2026.

Jens Eirik Saethre, Roberto Azevedo, and Christopher Schroers. Combining frame and GOP embeddings for neural video representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9253–9263, 2024.

Seungjun Shin, Suji Kim, and Dokwan Oh. Efficient neural video representation with temporally coherent modulation. In *European Conference on Computer Vision*, pp. 179–195, 2024.

Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

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.

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems*, volume 33, pp. 7537–7547, 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.

Alonso Urbano, David W. Romero, Max Zimmer, and Sebastian Pokutta. Neural field tokenizations with hierarchy and spatial locality priors. *arXiv preprint arXiv:2606.08204*, 2026.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

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.

Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004.

Thomas Wiegand, Gary J. Sullivan, Gisle Bjøntegaard, and Ajay Luthra. Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576, 2003.

Ian H. Witten, Radford M. Neal, and John G. Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, 1987.

Hao Yan, Zhihui Ke, Xiaobo Zhou, Tie Qiu, Xidong Shi, and Dadong Jiang. DS-NeRV: Implicit neural video representation with decomposed static and dynamic codes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 23019–23029, 2024.

Shuyi Zhang, Ke Liu, Jingjun Gu, Xiaoxu Cai, Zhihua Wang, Jiajun Bu, and Haishuai Wang. Attention beats linear for fast implicit neural representation generation. In *European Conference on Computer Vision*, pp. 1–18, 2024.

Jiancheng Zhao, Yifan Zhan, Qingtian Zhu, Mingze Ma, Muyao Niu, Zunian Wan, Xiang Ji, and Yinqiang Zheng. Tree-NeRV: Efficient non-uniform sampling for neural video representation via tree-structured feature grids. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15076–15085, 2025.

Qi Zhao, M. Salman Asif, and Zhan Ma. DNeRV: Modeling inherent dynamics via difference neural representation for videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2031–2040, 2023.

Qi Zhao, M. Salman Asif, and Zhan Ma. PNeRV: Enhancing spatial consistency via pyramidal neural representation for videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19103–19112, 2024.

SUPPLEMENTARY MATERIAL

The supplement provides implementation details and supporting evidence for the main-paper comparisons: state organization, storage allocation, and local rendering. Additional examples delimit the reconstruction and access claims rather than introduce separate downstream tasks.

A IMPLEMENTATION AND EVALUATION

A.1 MODEL AND TRAINING

The implementation settings in Table 4 apply to the main PAIRNeRV models. F denotes clip length, A/I denotes appearance/innovation dimensions, and E20/E150 denote training epochs. The state ceiling is 884,736 bits per clip; shared model weights are excluded.

Table 4: **Implementation settings.** The allocation changes transmitted code dimensions, not the patch layout.

Component	Setting
Input	Consecutive head-sampled frames; 256×256, YUV
Tokenizer	16×16 convolution, stride 16, no padding; width 768
Global encoder	Five pre-LN layers (Ba et al., 2016); six heads of width 64; FFN width 3712, GELU (Hendrycks & Gimpel, 2016); outer tokenizer residual
Appearance writer	One query-decoder layer; width 768, six heads of width 64, FFN width 3712; four-frame pooling groups
Innovation writer	Shared affine projection per frame–patch feature
Renderer	Two-layer MLP; hidden width 144, SiLU (Elfwing et al., 2018), output offset 0.5
Geometry	Patch embedding width 32; local coordinates and eight log-spaced Fourier frequencies from 1 to 32 per axis
Optimizer	AdamW (Loshchilov & Hutter, 2019); learning rate 10^{-4}; weight decay 10^{-2}
Training	Batch size two; FP16 AMP; learning rate 10^{-5} from E135 for E150 models
Storage	Continuous-state training; signed Q8 after training

Configurations. PAIRNeRV uses seed 42 in Table 1. Its selected E150 profiles are A74/I88, A66/I45, and A74/I22 at F4/F8/F16. The E20 F8, readout controls, and paired-storage comparison retain A90/I42. Table 2 summarizes seeds 42–44.

A.2 COMPARISON MODELS AND RESULT SOURCES

Baseline reconstruction metrics in Table 1 follow the published comparison in CoANeRV (Guo et al., 2026). Table 3 uses our native CoANeRV and PAIRNeRV implementations trained at seed 42 and evaluated under a common protocol.

Native CoANeRV retains its own architecture. The CoANeRV-style execution control instead uses five encoder layers and one decoder layer, while the matched A/I readers retain PAIRNeRV's writers and renderer. These controls answer different questions and are not interchangeable CoANeRV results.

The local N384 FP32 K400 scores are 32.77, 29.57, and 26.90 dB at F4/F8/F16, below the published 34.8, 32.1, and 30.9 dB. The available comparisons do not isolate the source of these discrepancies; we do not attribute them to seed variation or a single precision choice. Table 1 retains the published reference, whereas Table 3 measures capacity and storage effects within the local implementations. Neither comparison alone isolates state organization; the matched readers address that narrower question.

Storage and computation precision. FP32 and Q8 label storage, not network arithmetic. Main reconstruction and allocation results use AMP evaluation. Reader and paired-storage quality uses

FP32 computation with TF32 disabled; each storage pair uses identical inputs and arithmetic. Timings use the matched execution settings specified in Appendix E. Neither changing evaluation arithmetic nor comparing different allocations isolates quantization loss.

A.3 Data and Metrics

We use the fixed K400 (Kay et al., 2017) training subset and filtered validation manifest of the main comparison. Videos are resized along the shorter side and center-cropped to 256×256, with consecutive frames sampled from the sequence head. Clips shorter than the requested frame count are excluded rather than padded. Table 5 lists the eligible evaluation counts. K400 in the main tables denotes held-out K400 validation, not an additional official test split. We use the V2 release of Something-Something (Goyal et al., 2017) and UCF101 (Soomro et al., 2012); neither receives target-domain optimization nor target-domain checkpoint selection.

For PAIRNeRV, Train denotes evaluation on the training videos after optimization, with no parameter updates. Within each PAIRNeRV row of Table 1, the reported splits use the same trained model and Q8 state. Online training averages are not used.

Table 5: **Evaluation coverage.** Counts are eligible videos; all storage pairs share the same ordered inputs.

Split	F4	F8	F16
K400 Train	10,000	10,000	9,997
K400 validation	13,732	13,732	13,731
SthV2	20,000	20,000	19,947
UCF101	3,500	3,500	3,500

The short-clip comparisons follow the baseline's normalized YUV convention: PSNR uses equal channel weights and unit peak, and SSIM (Wang et al., 2004) uses the same unclamped predictions. PSNR is computed from each ordered batch of two clips and averaged with clip-count weights. This differs from averaging per-video PSNR or converting globally pooled MSE to dB. SSIM in the reader and paired-storage tables uses the same five-dimensional layout; framewise evaluation agrees to numerical precision on the paired storage tests. High-resolution studies specify their RGB/YUV metric space separately in Appendix F.

A.4 Configuration Selection

The main-table profiles remain fixed across datasets. The expanded allocations are the observed K400-best profiles from the full-training sweep: A74/I88, A66/I45, and A74/I22. The eight-frame configuration matches the E150 reconstruction row in Table 1; paired precision and reader controls retain their separately evaluated A90/I42 allocation. Compact controls use A20/I22, A20/I11, and A12/I6 without an exhaustive allocation search.

K400 validation is used for allocation selection as well as reporting; its reported scores are not an independent test of that selection. SthV2 and UCF101 assess transfer of the fixed configurations without target-domain selection. Dataset-specific maxima in the allocation curves are visual references, not separate choices of a model for each dataset.

A.5 Parameter and Training-Cost Accounting

Encoder size includes tokenization, embeddings, the context Transformer, appearance queries, and both writers. INR size counts learned modules from recovered state to reconstruction, including complete projection matrices and their masked capacity; unused checkpoint-compatibility modules are excluded. Total shared sizes are 44.915M, 44.957M, and 44.935M at F4/F8/F16. Renderer widths and counts are detailed in Appendix B.3.

Baseline INR sizes and training totals in Table 1 are published values. PAIRNeRV GPU-hours are short-run A100 estimates, not elapsed times of the AMP quality runs: three no-AMP, TF32-enabled epochs provide startup and steady-state costs, extrapolated with validation every ten epochs and checkpoint writing. Queue time, allocation search, and cross-dataset evaluation are excluded.

F8 timing uses the fixed A90/I42 profile. These estimates are resource accounting, not a matched training speed comparison; Tables 2 and 3 instead report isolated batch-two step times.

Baseline state counts convert FP32 values to bits: 24,960 for TransINR, 25,600 for GINR, 384×72 for CoANeRV, and FastNeRV's rounded published 24.1k count. PAIRNeRV stores 109,056 Q8 values plus 36 metadata bits. These are representation budgets, distinct from actual file rates.

B State Formation and Reusable Rendering

B.1 Encoder and Token Writers

Let $\mathbf{Z} = \mathrm{Tok}_\phi(V) \in \mathbb{R}^{T\times P\times d}$ denote the raw tokenizer features. Spatial and frame-time embeddings form $\mathbf{Z}^+_{t,p} = \mathbf{Z}_{t,p} + \mathbf{e}^{\mathrm{space}}_p + \mathbf{e}^{\mathrm{time}}_t$, where $\mathbf{e}^{\mathrm{time}}_t = \mathrm{Time}_\phi((t-1)/\max(T-1,1))$. The implementation normalizes zero-based indices; the paper uses one-based indices. Its auxiliary token $\mathbf{c} = \mathrm{Time}_\phi(0) + \mathbf{e}^{\mathrm{aux}}$ participates in attention but is discarded before writing state. With the five-layer Transformer $\mathcal{T}_\phi$,

$$\mathbf{H} = \mathrm{Patch}\big(\mathcal{T}_\phi([\mathbf{Z}^+; \mathbf{c}])\big) + \mathbf{Z}.$$

Patch discards the auxiliary output and restores the frame–patch layout. Sublayers use pre-normalization and residual connections; the outer residual adds raw, not position-embedded, tokenizer features.

Consecutive groups $\mathcal{G}_g$ of four frames form appearance memory $\overline{\mathbf{H}}_{g,p} = |\mathcal{G}_g|^{-1} \sum_{t\in\mathcal{G}_g} \mathbf{H}_{t,p}$. All GP entries supply keys and values to P shared learned queries. Cross-attention, self-attention, and FFN sublayers precede layer normalization, a shared affine projection, and learned slot-wise scales and offsets. F4 uses one temporal mean; longer clips retain multiple groups. Every output slot still supplies one appearance code for the entire clip. Innovation is a direct projection of $\mathbf{H}_{t,p}$, not a residual subtraction.

Ownership describes each code's output address, not a local mask on the encoder or appearance writer. The grid is fixed in image coordinates and does not track objects. Both families may encode texture, geometry, and motion-related content. Within an allocation cohort, inactive suffixes of fixed-width projections are zero-masked and omitted from storage; recovered codes are zero-padded before projection.

B.2 Projection Reuse

Partitioning the first renderer matrix gives

$$\mathbf{h}_{t,p,\boldsymbol{\xi}} = W_a \widetilde{\mathbf{A}}_p + W_i \widetilde{\mathbf{I}}_{t,p} + W_p \mathbf{e}_p + W_\xi \ell(\boldsymbol{\xi}) + \mathbf{b}.$$

The implementation stores reusable projections separately, rather than a dense per-pixel static sum. Appearance projections belong to one clip; geometry depends only on the model and output grid. Innovation projection is computed once per frame–patch pair and shared by its pixels. The MLP tail, including its nonlinear activation and fixed color offset, still runs for each output pixel. Reuse preserves the function in exact arithmetic, not necessarily bitwise in floating point. Numerical checks and first-use costs are specified in Appendix E.2.

B.3 Renderer Parameters

The learned renderer comprises A/I expansion, patch embeddings, and the MLP, including biases and complete fixed projection matrices. Coordinate encoding and dequantization add no parameters. F4 recovers width-79 features; F8/F16 recover width 123, all with MLP hidden width 144. F8 and F16 differ in innovation input width, 50 versus 26, accounting for $(50-26)\times 123 = 2{,}952$ parameters. The resulting counts are 54,852/75,101/72,149. Their non-monotonicity reflects architecture widths, not frame-specific decoder weights or the number of stored clips.

C Readout Controls and Intervention Scope

C.1 Matched Readout Controls

The global A/I controls in Table 2 both return two vectors and retain the writers, parameter shapes, and 872,484-bit state. Joint readout applies two queries to the union of appearance and current-frame innovation banks; separate readout applies one query to each bank. For a bank $\mathcal{B}$, the content-addressed output is

$$R_p(\mathcal{B}) = \sum_{\mathbf{z}\in\mathcal{B}} \frac{\exp(\sqrt{h}\,\bar{\mathbf{q}}_p^\top \bar{\mathbf{z}})}{\sum_{\mathbf{z}'\in\mathcal{B}} \exp(\sqrt{h}\,\bar{\mathbf{q}}_p^\top \bar{\mathbf{z}}')}\mathbf{z}.$$

Bars denote L2 normalization with an $\epsilon = 10^{-6}$ floor. The learned patch embedding is repeated and truncated to width h; its one-channel circular shift supplies the second query. No additional Q/K/V matrices are introduced. Attention arithmetic uses FP32 before conversion to the renderer dtype. Local lookup directly indexes the corresponding pair. These controls isolate the specified retrieval interfaces, not all possible global-attention parameterizations.

All-I instead stores 53 innovation values per frame–patch and no appearance stream, totaling 868,372 bits. It retains appearance-writer computation but discards its output, so it tests storage allocation rather than the saving from deleting that module. The main table's All-I result on SthV2 prevents a uniform claim that adding appearance improves every setting.

C.2 Stability Across Training Seeds

Table 2 uses FP32 evaluation with TF32 disabled, Q8 state, and the same ordered batch-two inputs. Train uses all 10,000 training videos; K400, SthV2, and UCF101 use 13,732, 20,000, and 3,500 videos, respectively.

Table 6 measures each change within the same training seed before summarizing seeds 42–44. Separating the two global reads improves PSNR for every seed on all four datasets, as does replacing separate global retrieval with local lookup. The corresponding SSIM differences are also positive in every case. These results support the two readout changes across the tested seeds; they do not require every row of Table 2 to improve monotonically, since All-I also changes how the storage budget is allocated. The standard deviations describe training-seed variation, not per-video confidence intervals; these two contrasts do not test interaction between role separation and locality.

Table 6: **Paired readout effects.** Within-seed differences are summarized as mean ± sample standard deviation across seeds 42–44. SSIM units are 10^{-3}; letters refer to Table 2.

Change	Metric	Train	K400	SthV2	UCF101
Separate − joint (d−c)	PSNR	0.370 ± 0.190	0.359 ± 0.181	0.335 ± 0.151	0.474 ± 0.250
	SSIM	3.412 ± 1.391	3.504 ± 1.434	2.607 ± 1.200	3.736 ± 1.494
Local − separate (e−d)	PSNR	0.243 ± 0.045	0.239 ± 0.046	0.200 ± 0.084	0.316 ± 0.056
	SSIM	2.299 ± 0.705	2.378 ± 0.703	1.526 ± 0.733	2.574 ± 0.793

C.3 Timing the Main Ablations

Table 2 uses seed-43 E150 checkpoints on an A100, with FP16 AMP and TF32 disabled. Four fixed K400 clips form two batch-two training inputs; decoding uses the first clip of each pair. After five warm-ups, three trials of 30 calls give six input/trial medians whose median is reported. A training step includes encoding, rendering, L2 loss, backward, and the AdamW/GradScaler update after restoring training state, excluding I/O, metrics, validation, and saving. It is not divided by batch size.

Cold decoding starts from GPU-resident Q8 state and rebuilds state-dependent and geometry caches on each call. It excludes file I/O and entropy decoding. Reuse off/on retains identical inputs and weights; the global readers do not implement this projection-cache path. For PAIRNeRV, cold latency is 2.62/2.65 ms with reuse off/on, so reuse is not a cold full-clip speedup. The repeated-frame setting in Appendix E.2 measures a different opportunity for amortization.

C.4 What the Interventions Establish

Figure 1(c) edits recovered codes while holding all other codes and weights fixed, then refreshes the affected caches. Appearance edits address one patch across frames; innovation edits address one frame–patch pair. No encoder input is masked and no output pixels are copied. Numerical checks at F4/F8 cover one-, four-, and sixteen-patch edits: change outside the edited spatial region is zero, and innovation edits leave other frames unchanged. Zeroing creates out-of-distribution codes; the test demonstrates readout scope, not semantic disentanglement. Input edits before global encoding, or edits that recompute a family-wide quantization scale, need not remain local.

D Storage and Allocation Details

D.1 Quantization and Bit Accounting

States are trained continuously. Each active code family is quantized independently per video. For a nonzero family vector $\mathbf{x}$, signed Q8 uses an FP16 scale and reconstructs

$$s = \mathrm{FP16}(\|\mathbf{x}\|_\infty/127)\,, \qquad \widehat{\mathbf{x}} = s\ \mathrm{clip}\left(\mathrm{round}(\mathbf{x}/s), -127, 127\right).$$

Inactive projection suffixes remain zero. With $P = 256$ spatial patches, the expanded states in the main paper occupy

$$B(T, d_a, d_i) = 8P(d_a + Td_i) + 36 = 872{,}484 \text{ bits},$$

where $d_a + Td_i = 426$ and metadata consists of two FP16 scales and a four-bit profile identifier. Appearance is charged once per patch; innovation is charged once per frame–patch. No entropy gain is assumed. The self-describing file format used for access measurements has additional headers (Appendix E.3).

D.2 Configurations for the Capacity Comparison

Table 3 compares independently trained E150 seed-42 models. Compact PAIRNeRV uses A20/I22, A20/I11, and A12/I6 at F4/F8/F16; expanded PAIRNeRV uses A74/I88, A90/I42, and A74/I22. Compact profiles contain 27,648 values and expanded profiles 109,056. FP32/Q8 storage requires 884,736/221,216 bits for compact state and 3,489,796/872,484 bits for expanded state, including their respective metadata. Each FP32/Q8 pair uses one checkpoint with identical FP32, TF32-off inference. Between compact and expanded models, differences also reflect retrained capacity and allocation, not quantization alone.

Native CoANeRV (Guo et al., 2026) retains its own architecture with $N = 384$ or 1535 tokens of width 72. Its per-video affine Q8 quantizer stores 32 metadata bits. The two banks require 884,736/221,216 and 3,536,640/884,192 bits in FP32/Q8, respectively. Thus the larger Q8 bank fits the smaller FP32 bank's ceiling. Published and local result sources are distinguished in Appendix A.2. The cross-method comparison retains different architectures; the internal reader controls separately test readout organization.

Training-step measurements for Table 3 run sequentially on one isolated A100 with batch two, FP16 AMP, and TF32 off. Two fixed input batches each undergo three 30-step windows after warmup; the result is the mean of 180 synchronized steps. Encoding, reconstruction, loss, backward, and optimizer update are included; input transfer, metrics, and saving are excluded. Each implementation retains its renderer and activation checkpointing. E150 optimizer states are restored, with probe learning rate 10^{-5} for all models; source checkpoints are unchanged.

D.3 Matched-Budget Capacity Across Seeds

Lower storage precision can also benefit an unstructured global token bank. To test this alternative use of the same budget, we retain the native CoANeRV (Guo et al., 2026) architecture and increase its number of 72-dimensional video tokens from $N = 384$ to $N = 1535$. Both capacities are trained independently for 150 epochs on eight-frame K400 clips, with batch size two, head sampling, YUV inputs, and learning-rate decay at epoch 135. We repeat both with seeds 42, 43, and 44; neither

Table 7: **Matched-budget capacity comparison.** PSNR is mean ± sample standard deviation over three seeds; SSIM is the mean. Underlined payloads identify the common-ceiling comparison, while the larger FP32 CoANeRV state exceeds that ceiling. Expanded Q8 PAIRNeRV retains higher quality at the common ceiling.

Model / storage	Payload (kbit)	PSNR (dB)				SSIM			
		Train	K400	SthV2	UCF101	Train	K400	SthV2	UCF101
CoANeRV N=384, Q8	221.216	32.333 ± 0.575	29.480 ± 0.876	31.171 ± 0.832	30.616 ± 1.008	0.93157	0.91258	0.92378	0.92433
CoANeRV N=384, FP32	884.736	32.355 ± 0.586	29.490 ± 0.882	31.183 ± 0.840	30.629 ± 1.016	0.93178	0.91277	0.92392	0.92460
CoANeRV N=1535, Q8	884.192	35.768 ± 0.952	34.089 ± 1.468	36.054 ± 1.427	35.591 ± 1.750	0.96084	0.95395	0.95883	0.96224
CoANeRV N=1535, FP32	3536.640	35.817 ± 0.980	34.123 ± 1.490	36.097 ± 1.456	35.635 ± 1.779	0.96113	0.95423	0.95907	0.96257
PAIRNeRV, Q8	872.484	37.045 ± 0.023	36.886 ± 0.024	39.082 ± 0.053	38.963 ± 0.039	0.96386	0.96232	0.96627	0.97006

training path quantizes the state. PAIRNeRV uses the A90/I42 allocation and the same three seeds, rather than Table 1's selected A66/I45 profile.

Table 7 uses a common evaluator for both methods: FP32 computation with TF32 disabled, the complete four data lists, and the YUV aggregation specified in Appendix A. Uncertainty is the sample standard deviation across the three seeds. Quantization changes native CoANeRV's mean PSNR by less than 0.05 dB at either capacity. Expanding the bank while retaining the bit ceiling improves mean PSNR by 3.41, 4.60, 4.87, and 4.96 dB on Train, K400, SthV2, and UCF101, respectively. Saved bits therefore help the global-token baseline as well as PAIRNeRV.

Under this common evaluator, PAIRNeRV exceeds expanded CoANeRV Q8 by 1.28, 2.80, 3.03, and 3.37 dB on the same splits, with higher mean SSIM and a slightly smaller state. This comparison supports an advantage beyond leaving the baseline at its original token count, but it does not isolate token organization: the methods retain their own encoders, writers, renderers, and parameter counts. The matched readers in Table 2 address that narrower question. The larger CoANeRV seed variation is retained in the table, and this local comparison does not resolve the published-versus-reproduced differences reported in Appendix A.2.

D.4 Allocation Curves

Figure 7 complements the main relative curves with absolute PSNR. The F4/F8/F16 sweeps contain 9/13/8 allocations, each trained to E150 with seed 42 and E135 learning-rate decay, then evaluated on all four data splits using Q8 and AMP. Every allocation occupies 872,484 bits. The sweep enumerates positive integer solutions of $d_a + Td_i = 426$ within fixed projection widths, not all possible architectures. Model parameter counts remain fixed within each clip-length cohort.

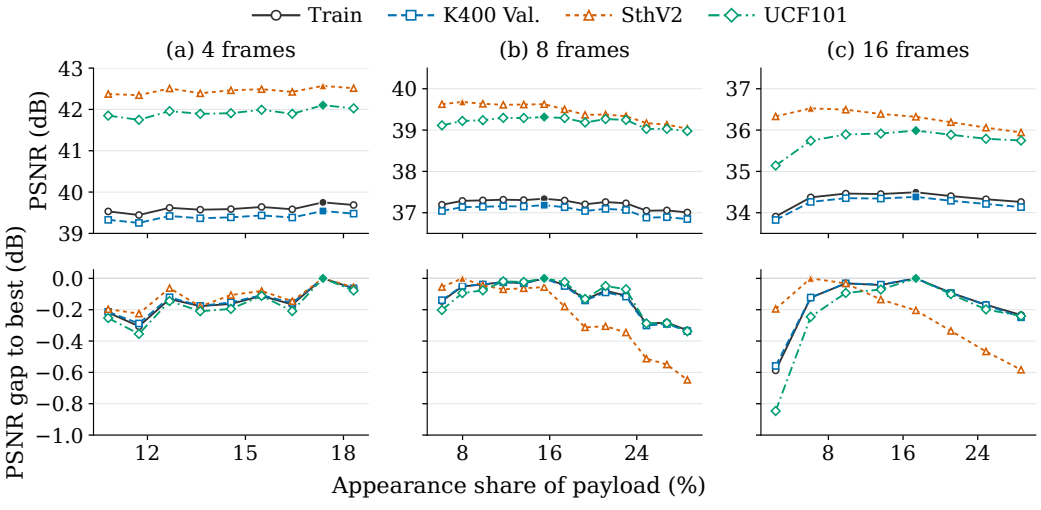


Figure 7: **Appearance–innovation allocation across datasets.** Columns separate clip lengths. Top panels show absolute PSNR; bottom panels subtract each curve's best measured PSNR, marked by a filled symbol. No single appearance share leads every setting.

The relative view subtracts each curve's largest measured value; it preserves within-curve differences but does not compare absolute quality across datasets. Configuration selection is specified in Appendix A.4. The broad high-quality regions and differing dataset preferences support an allocation trade-off, not a seed-invariant integer optimum or a claim that either role should receive all available capacity.

D.5 Allocation Stability Across Training Seeds

The single-seed curves describe allocation sensitivity, not a seed-invariant ranking. Their dataset-specific maxima are visual references; the main-table profiles remain fixed across datasets. These controls cover selected profiles and their neighbors, not the entire grid in Figure 5.

Repeating the selected profiles under the final schedule gives PSNR sample standard deviations of 0.097, 0.040, and 0.031 dB for F4, F8, and F16 (Table 8). These repetitions characterize the selected endpoints. For the neighboring F4 profile A66/I90, the difference from A74/I88 is $+0.0437$ dB at seed 43 and -0.0361 dB at seed 44, with a matched two-seed mean of $+0.0038$ dB. The small differences and reversal support a favorable local region with seed-dependent ordering.

Table 8: **Allocation variation across seeds.** Values are mean $\pm$ sample standard deviation. The selected allocations show small quality variation across the repeated runs.

F	Allocation	PSNR ↑	SSIM ↑
4	A74/I88	39.4 ± 0.10	0.9761 ± 0.0003
8	A90/I42	37.0 ± 0.04	0.9652 ± 0.0001
16	A74/I22	34.4 ± 0.03	0.9461 ± 0.0002

This control reports E150 K400 validation with Q8 state and AMP evaluation. Its F8 allocation is A90/I42, not the A66/I45 profile subsequently selected for Table 1. It does not establish seed stability for every allocation in the complete sweep.

E Rendering and Access Protocols

The efficiency results separate GPU-resident decoding, repeated-frame reuse, and file-to-region access. All comparisons below use an A100-SXM4-80GB; their different timing boundaries are not pooled into an end-to-end codec speedup. Main-table ablation timings are specified in Appendix C.3.

E.1 Resident-State Throughput

Figure 6(a) compares E150 F8 models on state encoded from eight real K400 head clips. PAIRNeRV uses A90/I42 in Q8; native CoANeRV uses N384 in FP32. Both use AMP inference with TF32 off. Larger batches repeat the input pool. Ten warm-ups precede three trials of twenty calls; FPS uses the median CUDA trial-mean latency. Encoding, recovery, and file I/O are excluded. First-use latency is instead the median synchronized wall time over sixty first decodes of newly encoded resident states.

CoANeRV's native 64-pixel execution block incurs small-batch overhead. An output-checked 256-pixel block raises batch-one throughput from 413 to 1,764 FPS, while both approach 1,975 FPS at batch 32. The main figure uses the faster setting. PAIRNeRV's 3.0–3.8$\times$ warm throughput advantage therefore does not rely on the slower baseline schedule; its first-use batch-one advantage is smaller, $1.95\times$.

E.2 Cross-Frame Reuse

Figure 6(b) uses identical F8 weights and inputs with reuse off/on, FP32 computation, and TF32 disabled. Eight K400 clips supply distinct-frame requests. New-clip preparation and the first appearance projection are included; geometry is already resident. Each input contributes the median cumulative time over five repetitions, and the curve reports the median across inputs divided by requested frame count. Clearing geometry adds first-request cost and can remove early gains.

Appearance is clip-specific; geometry depends on the model and grid. Innovation and the nonlinear tail remain per-request computation. The eight-request latency changes from 1.17 to 1.06 ms per frame. This amortization differs from cold, vectorized full-clip decoding, where projections are already shared within the call and Table 2 shows no corresponding speedup. The strict-FP32 sequential checks pass maximum absolute error 10^{-3}, MSE 10^{-8}, and absolute PSNR change 0.01 dB. This does not assert bitwise equivalence for arbitrary mixed-precision regrouping.

E.3 Stored-State Access

The fixed-width Q8 file contains 109,056 code bytes and a 48-byte self-describing header, for 109,104 bytes in total. The header records version, shapes, scales, and family offsets beyond the minimal representation ledger. This indexed file is not an entropy stream; rates from an entropy-coded variant cannot be combined with its access times.

Figure 6(c,d) uses the F8 E150 seed-42 model with FP32 computation and TF32 off. Three UVG sequences supply 256×256 center crops over 32 frames. Frames 0, 16, and 31 provide nine video–frame cases, each requesting 1, 4, 16, 64, or 256 centered 16×16 patches. The same Q8 files support indexed code reads, whole-state reads followed by local rendering, and full-frame rendering followed by cropping.

Each trial includes file reading, parsing, transfer, dequantization, and first appearance projection. Model weights, geometry, and OS file pages are warm, while video-state caches are cleared between trials. Each case uses twenty trials after an initial call; lines and bands show the median and interquartile range across case medians. Logical bytes are application reads, not physical disk traffic. Indexed reads require $48 + 132n$ bytes for n patches, versus 109,104 for either whole-state path.

All 225 output checks pass maximum absolute error 10^{-4} and MSE 10^{-10} against full-frame reconstruction. Fewer bytes do not guarantee less latency: indexed access is faster than full-frame rendering only for one- and four-patch requests, and its one-patch latency overlaps the whole-state local path. The result identifies when selective access helps, not a uniform speedup for all region sizes.

E.4 High-Resolution Region Requests

UVG (Mercat et al., 2020) Beauty and Jockey supply 1080p frames, and MCL-JCV (Wang et al., 2016) SRC11 and SRC23 supply 720p frames. Each provides sixteen requests across two F8 clips for 1, 4, or 16 dispersed 64×64 regions, either stationary or translated in 16-pixel steps. Requested-tile and full-frame paths batch up to eight 256×256 tiles; indexed access processes required ownership patches tile by tile.

All paths include file reading through output gathering. Weights, geometry, and file pages are warm, but no video state persists across requests. Five sequence repetitions follow warmup; entries summarize the four video–motion case medians per resolution. All 72 cases meet the same 10^{-4} maximum error and 10^{-10} MSE tolerances.

Table 9: **High-resolution region access.** Milliseconds per request for identical output regions. Batched tile rendering remains effective as requests become fragmented; patch indexing need not.

Resolution	Regions	Indexed patches	Requested tiles	Full frame
1080p	1	1.78	1.68	22.24
1080p	4	7.43	3.56	22.27
1080p	16	31.07	11.38	22.32
720p	1	1.75	1.66	8.97
720p	4	5.30	2.68	9.00
720p	16	21.07	7.90	9.08

The appropriate execution granularity depends on request size and fragmentation. These are comparisons within one decoder, not an inter-method access-speed ranking or measurements of cold physical storage.

F High-Resolution Evidence and Limitations

F.1 Matched-Window Reconstruction

The high-resolution comparisons use UVG (Mercat et al., 2020) at 1920×1080 and local 1280×720 copies of MCL-JCV (Wang et al., 2016), denoted MCL720. A 256×256 input tile contains 16×16 ownership patches, each 16×16 pixels. Global encoding gathers context within the tile-clip, not the entire high-resolution video. Independently encoded tiles are reassembled with overlap averaging at frame boundaries.

Table 10 evaluates the same first 128 frames from all seven UVG sequences and sixteen eligible MCL identities, with sixteen F8 windows per sequence. The MCL set is 01–04, 10–16, 19, 26–28, and 30; source 03 uses the second local version. The same common eight-bit RGB targets are recovered from the existing full-range YUV preprocessing by inverse conversion, clipping, and rounding. Predictions are converted to RGB and clipped before pooled-pixel PSNR and frame-mean SSIM.

Both models use seed-42 E150 checkpoints, AMP inference, and 256×256 tiles. PAIRNeRV stores A90/I42 in Q8; native CoANeRV stores N384 in FP32. The representation rate counts all tile-window states per valid target pixel, excluding shared weights and container overhead. This is a common-input method comparison, not a codec benchmark or an isolated architecture ablation.

Table 10: **Matched-window high-resolution reconstruction.** Both methods reconstruct identical F8 windows. Rates count representation state, not entropy-coded files; CoANeRV is the local reproduction.

		UVG 1080p			MCL720		
Method	State	bpp_{repr}	PSNR	SSIM	bpp_{repr}	PSNR	SSIM
CoANeRV	FP32	2.133	27.8	.804	1.800	22.9	.749
PAIRNeRV	Q8	2.104	36.5	.911	1.775	32.1	.918

F.2 Qualitative Protocol and Failure Cases

Figure 3 uses locally trained E150 models on common head-sampled F4 clips, with FP32 inference and TF32 off. The fourth frame and identical 64×64 crops are displayed after native-color-to-RGB conversion, clipping, and eight-bit rounding. Enlargement is nearest-neighbor, without enhancement. PAIRNeRV uses A74/I88 and Q8; FastNeRV uses one fixed posterior sample, not best-of-several selection. The K400 example is a diagnostic clip; SthV2/UCF101 crops were chosen from reference content and frame change before reconstruction.

For Figure 4, both F8 seed-42 E150 models reconstruct sixteen native-resolution frames spanning two clips, with the same tile layout and state settings as above. Reference changes and texture determine the crop positions independently of the reconstructed outputs. Neither qualitative figure estimates the frequency of success or failure.

Figure 8 retains difficult F8 A90/I42 examples from a twelve-case set selected for frame change, low PSNR, or weak appearance-only reconstruction within a 512-video diagnostic pool. Fine texture and small text remain degraded, including in low-change content. The comparison uses the same AMP continuation from E120 to E150 with E135 decay; improved average PSNR does not establish improved temporal fidelity for every sequence. These deliberately selected cases are not a prevalence estimate.

F.3 Scope of the Claims

Patch ownership constrains recovered-state readout, not the influence of input edits through the encoder. Temporal roles specify storage and access, not supervised separation of appearance and motion. The experiments support the tested state budgets and request patterns; they do not establish a universal advantage for global encoding, all appearance allocations, or indexed execution. Independent spatial tiles also limit cross-tile modeling. The following coded-file comparison separates compression from selective access and retains cases where PAIRNeRV is not advantageous.

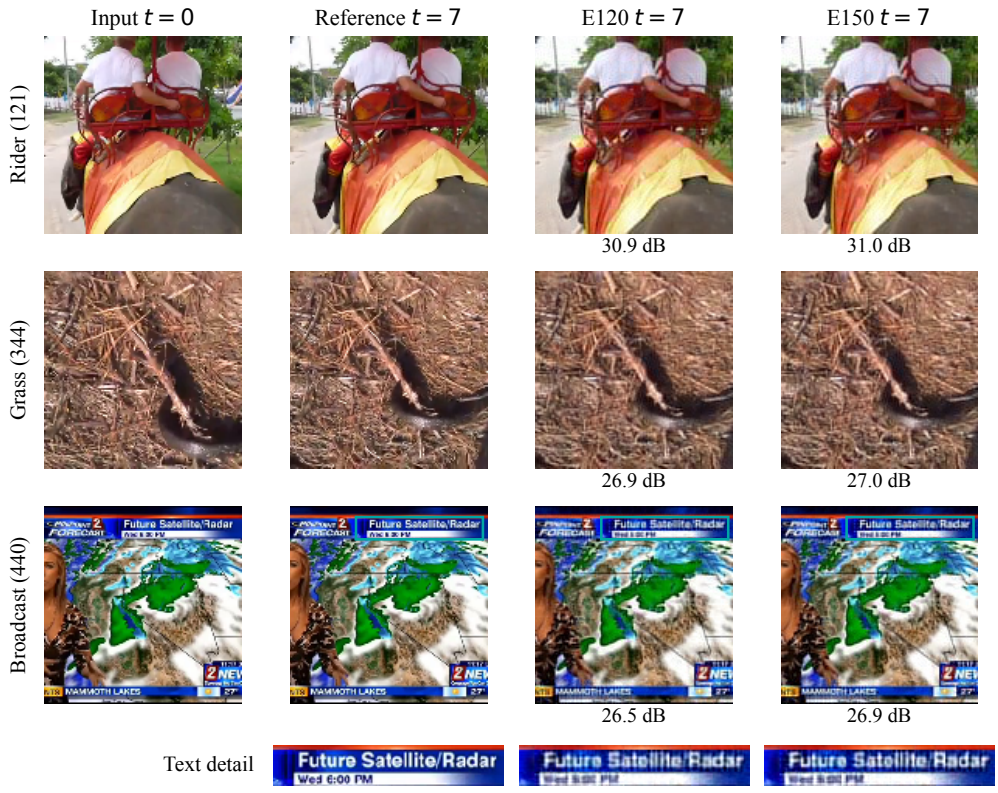


Figure 8: **Difficult reconstruction cases.** The first and last input frames show temporal change; E denotes the training epoch for two reconstructions of the same last frame. The bottom strip enlarges marked text without enhancement, and PSNR scores the full clip. Fine texture and text remain degraded after additional training; these selected cases do not estimate failure prevalence.

F.4 Coded Rate and Its Limits

Figure 9 compares actual file rates, including quantization metadata, headers, and padded tails; pretrained shared weights are excluded. All methods reconstruct 3,900 frames from seven UVG sources at 1080p and 4,115 frames from all 30 local MCL720 videos. Common eight-bit inputs are obtained by inverting the existing full-range YUV preprocessing, clipping, and rounding. After full-frame assembly, PSNR pools valid-pixel YUV errors equally across channels; SSIM averages full-frame scores. This is not native YUV420 or a 6:1:1 weighted metric.

Neural curves vary stored precision over Q3/Q4/Q6/Q8 without target-video fitting or QAT. PAIRNeRV uses Compact F8 A20/I11, E150 seed 42; native CoANeRV (Guo et al., 2026) uses N384, and CoANeRV-style is the separate five-layer-encoder, one-layer-writer control. FastNeRV (Chen et al., 2024) uses its F8 E150 model; TeCoNeRV (Padmanabhan et al., 2026) retains its official F8 model, quantizer, and closed-loop residual coder. TeCoNeRV tiles are 270×384; the other methods use 256×256. PAIRNeRV, CoANeRV, and FastNeRV use arithmetic coding (Witten et al., 1987) through torchac (Mentzer et al., 2019), storing symbol distributions and optionally predicting previous integers. References reset per video.

H.264/AVC (Wiegand et al., 2003), HEVC (Sullivan et al., 2012), and AV1 (Han et al., 2021) use FFmpeg x264/x265/libaom, respectively. x264/x265 use `veryslow` and CRF 18/24/30/36/42/48; libaom uses `cpu-used=4` and CRF 20/28/36/44/52/60. All use PSNR tuning. Input is converted to limited-range BT.709 (ITU-R, 2015) YUV420 and decoded back before the common metric transform. Each eight-frame, 256×256 block is coded independently, with no references across blocks; x264/x265 disable scene-cut insertion. Boundary-aligned windows produce 40 tiles per 1080p frame

and 15 per 720p frame. Overlaps are averaged; incomplete clips repeat the final frame. All duplicated, padded, and indexing bytes count toward rate, while only valid pixels enter the denominator.

These are aligned input windows, not identical access dependencies: neural paths retain cross-clip prediction, and TeCoNeRV retains different tile dimensions. The curves are adapted stored-state operating points rather than separately rate–distortion-trained codecs.

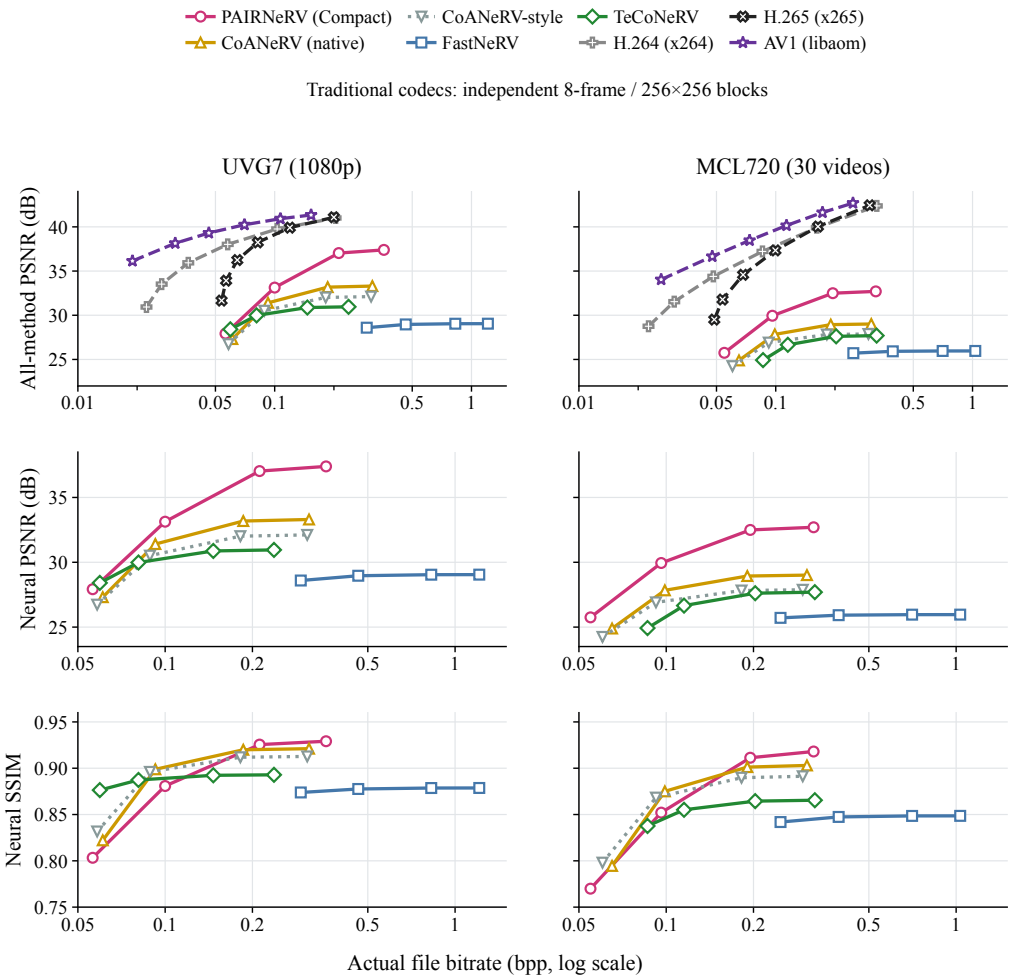


Figure 9: **Coded-file rate–distortion comparison.** Conventional codecs independently encode F8, 256×256 blocks. The lower rows enlarge the same neural points for PSNR and SSIM. Rates include actual file overhead; access dependencies are not identical.

PAIRNeRV's advantage over neural representations depends on precision and metric; lower-precision SSIM does not uniformly exceed native CoANeRV, and TeCoNeRV remains competitive at low rates. Conventional codecs retain a clear advantage: x265 reaches 39.91/37.34 dB at 0.1194/0.0996 bpp on UVG/MCL720, versus Compact Q6 PAIRNeRV's 37.03/32.49 dB at 0.2117/0.1949 bpp. Short-block coding therefore does not reverse that ranking. These entropy-coded files differ from the indexed Q8 files used for access timing; their rates and those latencies cannot be combined into a single compression–access claim.