OCTREE-BASED VIDEO REPRESENTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Video models commonly use uniform grids even though visual complexity varies substantially across space and time. We introduce OctVideo, which approximates a video clip with an octree. This hierarchy recursively partitions a spatio-temporal volume into eight subvolumes, so that smooth regions remain coarse while detailed regions receive finer cells. Each leaf stores local RGB values and spatio-temporal gradients, supplemented by a lightweight learned residual. For reconstruction, a Conv1D VAE maps the serialized cells to a regular latent grid and selectively refines details during decoding. Our VAE achieves 36.12 dB PSNR with 38.2M parameters and 189.4 GFLOPs per clip on Kinetics-400 (K400). It also generalizes zero-shot to the high-resolution Densely Annotated VIdeo Segmentation (DAVIS) 2016 dataset with reconstruction quality comparable to the best evaluated models. On both datasets, it requires the fewest model FLOPs and achieves the fastest encoding and decoding among the evaluated models. OctVideo also supports video understanding, achieving competitive recognition performance with few input tokens when trained from scratch. By exploiting the redundancy already present in video signals and efficiently processing sparse structures, OctVideo provides an efficient representation for video.

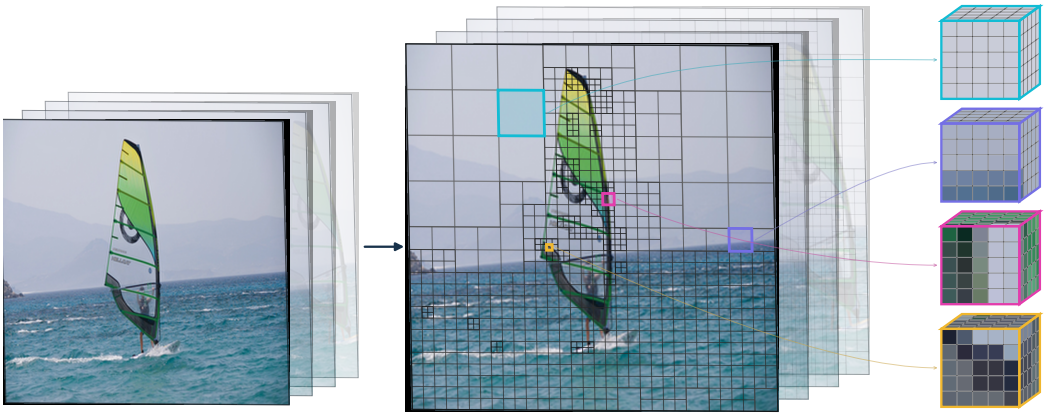

Figure 1: Content-adaptive video representation with OctVideo. A video volume is partitioned into large cells in smooth regions and finer cells where local approximation error is high. Colored boxes mark the cells enlarged on the right, illustrating different spatio-temporal scales.

1 INTRODUCTION

Video content varies in complexity across space and time, yet video models commonly process it on uniform grids during training and inference. A slowly varying background can occupy most of a clip, while fine detail is concentrated around object boundaries, textured surfaces, and transient events. Fixed-size tokens assign the same sampling density to these regions despite their different representational needs. An efficient video representation should allocate finer cells where the signal requires them, preserve a local approximation throughout the video, and expose enough structure for neural networks to process it efficiently.

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.

Existing approaches address different aspects of this problem. Classical and learned codecs remove redundancy for storage and transmission, but their compressed streams are not designed for direct neural processing (Wiegand et al., 2003; Lu et al., 2019). Video VAEs learn compact latent grids (Wan-Video, 2025; Kong et al., 2024), with efficient variants using wavelets and compressed sensing to reduce computation (Li et al., 2025; Cheng & Yuan, 2025). These architectures improve processing on regular grids without adapting input cell sizes to local signal complexity. For recognition, token pruning and merging reduce the computational cost by selecting or combining features, without explicitly retaining a locally reconstructible RGB signal (Hao et al., 2025; Bolya et al., 2023). Our goal is a compact input representation that combines content-adaptive allocation, direct reconstruction, and efficient neural processing.

Classical graphics methods have been using adaptive and hierarchical data structures to represent smooth regions coarsely and complex regions finely (Frisken et al., 2000; Ohtake et al., 2003). Learned octree and quadtree methods bring related principles to shapes and images (Wang et al., 2018; 2022; Wang, 2023; Ronen et al., 2023). Applying this idea to video poses two linked challenges: retaining local RGB variation within cells that span both space and time, and translating an irregular partition into computational savings in neural models.

In this paper, we introduce **OctVideo**, an octree-based representation that couples adaptive partitioning with local signal approximation. Each cell stores *a local RGB value and spatio-temporal gradients*, defining a first-order approximation within its bounds. The error of this approximation determines whether the cell is subdivided: smooth regions remain coarse, while regions with larger errors receive finer cells. The same local model thus determines both *what each cell stores* and *where the representation is refined*. The octree leaves cover the entire video volume, so every location has an explicit RGB approximation that can be evaluated without a learned decoder. For neural processing, a lightweight convolutional branch supplies learned residual features that enrich each leaf without changing the error-guided partition.

To turn this compact representation into efficient neural computation, we design a VAE around the octree structure. The encoder serializes mixed-depth leaves in Morton order and processes them with regular 1D convlutional kernels. Cell bounds then determine how encoded features are copied or pooled onto a regular latent grid. A hierarchical decoder learns from octree-derived refinement targets to predict which latent cells require finer processing. This design carries adaptive allocation from the input representation into neural encoding and decoding. The same representation supports recognition through patch-wise attention and sibling pooling along the octree hierarchy.

On Kinetics-400 (K400) (Carreira & Zisserman, 2017), the VAE takes averages approximately 3,700 adaptive cells as input, $17.9\times$ fewer than the 65,536 patches in the regular finest-level partition. Analytic reconstruction alone achieves 34.72 dB PSNR, demonstrating the fidelity of the representation before neural decoding. The learned VAE achieves 36.12 dB with 189.4 GFLOPs. Under the reported evaluation protocol, it decodes nearly two orders of magnitude faster than IV-VAE, the highest-PSNR baseline, with a 0.72 dB lower PSNR. Compared with the lightweight LeanVAE, OctVideo improves PSNR by 0.78 dB while using over 40% fewer model FLOPs and decoding $2.3\times$ faster. Zero-shot transfer to high-resolution DAVIS 2016 videos (Perazzi et al., 2016) yields 32.76 dB PSNR with a VAE runtime of 0.54 seconds per clip using spatial tiling. For recognition, training from scratch of OctVideo yields 80.1% K400 Top-1 accuracy with approximately 4,200 input tokens per view, within 0.1 percentage points of MViTv1-B using about $12\times$ fewer input tokens under the respective evaluation protocols. In summary, our contributions are:

- An adaptive video representation in which first-order RGB approximation guides octree refinement and preserves a directly reconstructible signal, with learned residual features for neural processing.
- An efficient VAE that connects mixed-depth cells to a regular latent grid and uses octree-supervised selective refinement to concentrate decoding computation where finer detail is needed.
- Experiments demonstrate substantial reductions in tokens, VAE compute, and decoding time while retaining reconstruction quality and recognition accuracy.

Our code is available in an anonymous repository.[1]

[1] `https://anonymous.4open.science/r/octvideo-release-A546`

2 Related Work

Classical video codecs remove statistical redundancy through motion-compensated prediction, block transforms, quantization, and entropy coding. Standards such as H.264/AVC and HEVC, progressively enrich the partition and prediction choices used for rate–distortion optimization (Wiegand et al., 2003; Sullivan et al., 2012). Learned codecs replace parts of this hand-designed pipeline with neural motion estimation and analysis–synthesis transforms (Lu et al., 2019; Agustsson et al., 2020). Both families are effective for storage and transmission, but their compressed streams are not designed as representations for direct neural processing: conventional streams must first resolve inter-frame prediction, while learned codecs remain specialized for pixel rate–distortion. OctVideo instead constructs an explicit spatio-temporal signal representation whose cells can be processed directly and also decoded locally.

Hierarchical local approximation underlies adaptive distance fields and partition-of-unity surfaces (Frisken et al., 2000; Ohtake et al., 2003). Adaptive O-CNN, dual octree graph networks, and OctFormer extend octree structures to learned shape representations and point-cloud processing (Wang et al., 2018; 2022; Wang, 2023), while Quadformer uses saliency-guided quadtrees for mixed-resolution image tokenization (Ronen et al., 2023). OctVideo applies adaptive local approximation to a dense RGB field over space and time. Its value-gradient descriptors support analytic reconstruction and provide the error criterion for refinement, while the resulting tokens and hierarchy support learned recognition and reconstruction.

VQ-VAE introduced discrete learned codes, which VideoGPT extended to video with a spatio-temporal encoder (van den Oord et al., 2017; Yan et al., 2021). Modern video generation systems more often use continuous latent grids: Wan, HunyuanVideo, CogVideoX, CV-VAE, and IV-VAE employ spatio-temporal encoders and decoders (Wan-Video, 2025; Kong et al., 2024; Yang et al., 2025; Zhao et al., 2024; Wu et al., 2025). WF-VAE and LeanVAE reduce their cost through wavelet processing, with LeanVAE additionally using compressed sensing (Li et al., 2025; Cheng & Yuan, 2025). More recent work further improves the latent representation itself: PyraTok learns language-aligned multi-scale codes, PV-VAE adds future prediction, and KATok performs content-dependent token dropping (Susladkar et al., 2026; Zhao et al., 2026; Lee et al., 2026). While these methods improve semantic utility or compression, their tokens do not directly decode the local signal. OctVideo instead adapts cell size while preserving full spatio-temporal coverage, refining only regions with high approximation error.

Efficient video architectures vary temporal sampling and feature resolution (Feichtenhofer et al., 2019; Fan et al., 2021; Li et al., 2022). Token reduction methods instead prune unimportant tokens, as in DynamicViT for images, select subsets for video recognition, as in LITE, or merge similar features, as in ToMe (Rao et al., 2021; Hao et al., 2025; Bolya et al., 2023). OneVision-Encoder uses codec information to select informative RGB regions (Tang et al., 2026). Pruning discards regions, while merging replaces distinct local signals with shared features. The resulting information loss can be acceptable for classification, but the retained tokens cannot faithfully reconstruct the input. OctVideo instead coarsens the signal itself: Smooth regions are represented with larger cells that store RGB values and gradients, whereas regions with higher approximation errors are refined into smaller cells. Every location therefore remains represented, enabling the same compact input to support both understanding and reconstruction.

3 Method

OctVideo allocates cells according to local approximation error, giving smooth regions a compact description and preserving finer structure where needed. We first describe the shared representation and efficient VAE for reconstruction, followed by the hierarchical classifier used for complementary recognition evaluation (Fig. 2).

3.1 OctVideo Representation

Let $V : \Omega \to \mathbb{R}^3$ be a video with coordinates $\mathbf{x} = (t, h, w)$. Each octree cell c covers a region Ω_c partitioned into a fixed $2 \times 4 \times 4$ grid of subregions $\Omega_{c,q}$, indexed by $q = 1, \ldots, 32$. The grid

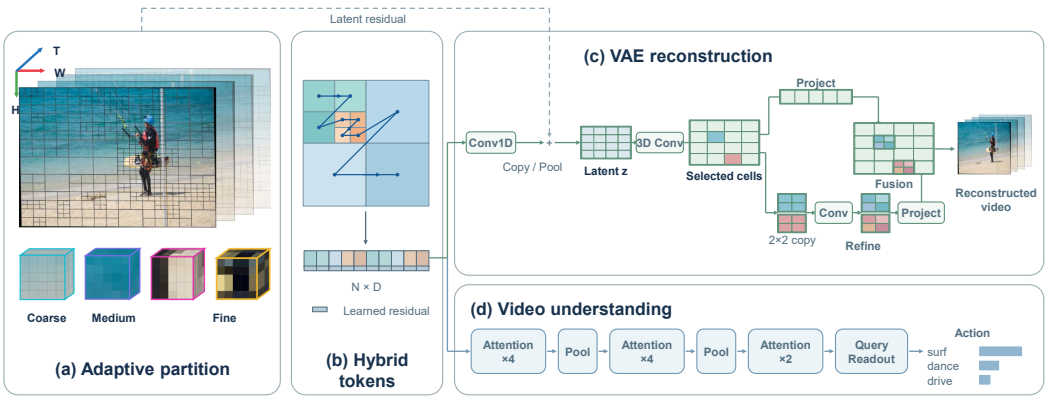

Figure 2: Overview of OctVideo. (a) Error-guided octree partitioning produces cells of different sizes, shown as equal-sized tokens for illustration. (b) Analytic descriptors and learned residual features form a Morton-ordered token sequence. (c) Conv1D encoding and copy/pool mapping produce a regular latent grid, supplemented by the dense latent residual shown as a dashed connection. The decoder refines selected cells and fuses them with coarse regions to reconstruct the video; 2D grids illustrate the 3D operations. (d) Hierarchical attention and sibling pooling support action recognition.

dimensions are fixed, while its video coverage varies with the cell's depth. Within each subregion,

$$\hat{V}_{c,q}(\mathbf{x}) = \mathbf{a}_{c,q} + \mathbf{G}_{c,q}\mathbf{u}_{c,q}(\mathbf{x}), \qquad \mathbf{x} \in \Omega_{c,q}. \tag{1}$$

Here $\mathbf{u}_{c,q}$ denotes centered local coordinates normalized to $[-\frac{1}{2}, \frac{1}{2}]$ along each axis (zero for a singleton axis). The RGB value $\mathbf{a}_{c,q} \in \mathbb{R}^3$ and gradients $\mathbf{G}_{c,q} \in \mathbb{R}^{3\times 3}$ with respect to these coordinates form a 12-dimensional descriptor, fitted by local least squares:

$$(\mathbf{a}_{c,q}, \mathbf{G}_{c,q}) = \arg\min_{\mathbf{a},\mathbf{G}} \sum_{\mathbf{x}\in\Omega_{c,q}} \|V(\mathbf{x}) - \mathbf{a} - \mathbf{G}\mathbf{u}_{c,q}(\mathbf{x})\|_2^2. \tag{2}$$

Starting from a coarse partition, we split a depth-d_c cell into eight children when $d_c < D$, where D is the finest depth, and

$$e(c) = \max_q \max_{\mathbf{x}\in\Omega_{c,q}} \|V(\mathbf{x}) - \hat{V}_{c,q}(\mathbf{x})\|_\infty > \tau_{d_c}. \tag{3}$$

Reconstruction and recognition use the same coarse-to-fine threshold schedule, $\boldsymbol{\tau} = (0.7, 0.8, 1.0)$, for the three successive refinement decisions. The tree construction procedure is detailed in Section A.4.

Each retained leaf becomes one token $\mathbf{f}_c = \mathrm{Flatten}([\mathbf{a}_{c,q}, \mathbf{G}_{c,q}]_{q=1}^{32}) \in \mathbb{R}^{384}$, regardless of its coverage. Together with its center $\mathbf{p}_c$, depth d_c, and bounds b_c specifying Ω_c, these tokens define the sparse representation $\mathcal{O}(V)$.

Learnable residual features. OctVideo determines the tree topology from local approximation error. A lightweight 3D convolutional branch extracts features $\mathbf{F}^0 = \psi_\theta(V)$ for all finest-level patches in parallel. We then aggregate these features according to the existing tree: for each leaf, we average and max-pool the features of the patches it covers, concatenate the two results, and project them to a learned residual $\mathbf{r}_c$. This adds learned detail without changing the partition. The encoder input is

$$\mathbf{x}_c^0 = \phi_{\mathrm{tok}}(\mathrm{LN}(\mathbf{f}_c)) + \mathbf{r}_c + \mathbf{e}_{d_c}, \tag{4}$$

where $\mathbf{e}_{d_c}$ encodes depth. The residual projection is zero-initialized so that it can gradually introduce complementary information without perturbing the original analytic token.

3.2 SEQUENCE-BASED VIDEO VAE

The VAE connects mixed-depth OctVideo cells to a compact regular latent grid. Its encoder mixes serialized leaf features and maps them to latent cells according to their spatio-temporal support. Its decoder starts from this grid and refines selected regions.

Serialization and sequence encoding. Sorting the leaves by Morton keys computed from their geometric centers defines a deterministic, locality-preserving order π for each video. We encode the resulting token sequence $\mathbf{S}^0 = [\mathbf{x}^0_{\pi(1)}, \ldots, \mathbf{x}^0_{\pi(N)}]$ using residual blocks:

$$\widetilde{\mathbf{S}}^\ell = \mathbf{S}^\ell + \mathrm{DWConv1D}_{k,\delta_\ell}(\mathrm{LN}(\mathbf{S}^\ell)), \tag{5}$$

$$\mathbf{S}^{\ell+1} = \widetilde{\mathbf{S}}^\ell + \mathrm{MLP}(\mathrm{LN}(\widetilde{\mathbf{S}}^\ell)). \tag{6}$$

Small dilations extend the receptive field along the sequence. Morton order retains coarse spatial organization, while bounds and depth preserve the geometry needed for later alignment. For fixed width and kernel size, these blocks have linear cost in the token count and use regular Conv1D and matrix multiplication kernels.

Mapping to the latent grid. We use $(4, 8, 8)$ temporal–spatial compression, giving an $8 \times 32 \times 32$ grid for a $32 \times 256 \times 256$ clip. After projecting encoded features to $\mathbf{h}_c$, we copy each coarse leaf's features to the latent cells it covers or average the features of the eight finest-level $(2, 4, 4)$ siblings within a latent cell:

$$\mathbf{g}_j = \begin{cases} \mathbf{h}_{o(j)}, & \text{if leaf } o(j) \text{ covers } j, \\ \frac{1}{8}\sum_{c \in \mathrm{child}(j)} \mathbf{h}_c, & \text{otherwise.} \end{cases} \tag{7}$$

The octree bounds determine this mapping directly. We add a latent positional embedding and a shallow RGB-to-latent residual to the resulting grid:

$$\mathbf{G}^0 = \mathbf{G}_{\mathrm{oct}} + \mathbf{P}_z + \mathbf{D}_r(V). \tag{8}$$

The residual branch uses a strided 3D convolution, one lightweight residual block, and a zero-initialized output projection. A shallow grid block mixes the combined features, and two heads then predict a diagonal Gaussian posterior. Reparameterization yields a latent representation with 16 channels per cell. This auxiliary path is deliberately lightweight: it adds only 30,849 parameters (0.081% of the 38.19M-parameter VAE) and 0.250 GFLOPs for a $32 \times 256 \times 256$ clip, accounting for less than 0.2% of the total VAE compute.

Selective hierarchical decoding. The decoder first processes the compact latent grid at $(4, 8, 8)$ compression with 3D convolutions. A split head selects cells for refinement; each selected cell is projected into eight $(2, 4, 4)$ child features with octant embeddings. A short Conv1D stack mixes these children in spatial order, concentrating finer-scale processing on the selected regions.

Each unsplit cell or refined child is mapped to local features covering its spatio-temporal region. These features are placed at their corresponding positions in a shared grid of size $T \times (H/2) \times (W/2)$, combining coarse and refined regions. A lightweight 3D convolutional head then doubles the spatial resolution and predicts the RGB video.

Training uses the input octree to supervise refinement: a latent cell is marked for splitting when it contains finest-depth leaves. We use these targets for teacher-forced expansion during training and predicted split masks during validation and inference.

Training objective. The VAE training objective combines an RGB reconstruction loss, an LPIPS perceptual loss, Gaussian KL regularization, and split supervision:

$$\mathcal{L} = \|\hat{V} - V\|_1 + \lambda_{\mathrm{lpips}}\mathcal{L}_{\mathrm{lpips}} + \lambda_{\mathrm{KL}}\mathcal{L}_{\mathrm{KL}} + \lambda_{\mathrm{split}}\mathcal{L}_{\mathrm{split}}. \tag{9}$$

Here $\mathcal{L}_{\mathrm{KL}}$ regularizes the posterior toward a standard normal prior, and $\mathcal{L}_{\mathrm{split}}$ is binary cross-entropy on the split logits. We set $\lambda_{\mathrm{lpips}} = 0.5$ and $\lambda_{\mathrm{split}} = 0.05$. We activate the per-leaf residual $\mathbf{r}_c$ after 10% of the total training epochs and the dense latent residual $\mathbf{D}_r$ after 20%. Their zero-initialized output projections make each transition continuous; after activation, both branches are optimized jointly with the complete VAE under $\mathcal{L}$. Full network and training specifications are provided in Sections A.2 and A.3.

3.3 Hierarchical Attention for Video Classification

The classifier groups each video's ordered token sequence into patches and applies self-attention within each patch, followed by feed-forward blocks. HWT rotary position encoding provides cell locations, and the depth embedding distinguishes cell extents. Between stages, complete groups of eight finest-level siblings are mean-pooled to their parent; shallower leaves pass through unchanged. The hierarchy therefore reduces the token count as features become more semantic, without padding the representation to a dense grid.

A single learned query then aggregates the final sparse memory $\mathbf{M}$:

$$\mathbf{g} = \mathrm{CrossAttn}(\mathbf{q}_{\mathrm{cls}}, \mathbf{M}, \mathbf{M}), \qquad \hat{\mathbf{y}} = \mathbf{W}_{\mathrm{cls}}\mathbf{g}. \tag{10}$$

The query is used only at readout. We train this branch with cross-entropy.

4 Experiments

We first examine the compactness and signal fidelity of the adaptive representation, then evaluate VAE reconstruction and the contributions of its design choices. Recognition provides a complementary test of semantic retention and learnability. The VAE uses MuonAdamW and the classifier uses AdamW; training details are provided in Section A.2.

4.1 Representation Analysis

Compactness and direct reconstruction. A regular finest-level partition contains 65,536 spatio-temporal patches per $32 \times 256 \times 256$ view. Under the VAE protocol, OctVideo averages 3,665 adaptive leaves on K400 validation, a $17.9\times$ reduction. Each leaf retains a local RGB approximation, allowing us to inspect the compressed signal before training a neural decoder. Direct reconstruction achieves 34.72 dB PSNR on K400 when PSNR is computed per clip and then averaged; details are provided in Section A.4. Qualitative comparisons in Appendix A.4 (Fig. 8) illustrate the contributions of explicit gradients and error-guided cell placement.

4.2 VAE Reconstruction

Our OctVideo VAE is trained from scratch for 50 epochs on the K400 training split using 32-frame 256×256 clips and MuonAdamW. Complete loss, optimizer, and learning-rate settings are provided in Section A.2. Our main evaluation of learned models tests whether the adaptive representation enables efficient video reconstruction. The comparison covers representative video VAEs and reports model size, reconstruction quality, inference memory usage, and measured encoding and decoding times. All experiments use 256×256 clips, $4 \times 8 \times 8$ compression, and 16 latent channels. Reported times measure VAE encoding and decoding; OctVideo construction takes less than 6.5 ms on an NVIDIA RTX 4090 and is reported separately.

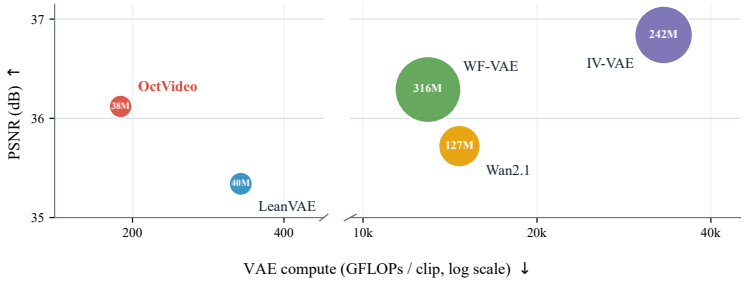

Figure 3: Kinetics-400 VAE quality–efficiency comparison. The logarithmic compute axis omits an empty range; bubble area represents parameter count.

Reconstruction quality and efficiency. Among the compared VAEs in Table 1, OctVideo uses the fewest parameters and requires the fewest FLOPs, with the shortest encoding and decoding times, while achieving high-quality reconstruction at 36.12 dB PSNR, 0.960 SSIM, and 0.0220 LPIPS.

Table 1: K400 video VAE comparison at 256^2 with $4 \times 8 \times 8$ compression and 16 latent channels. T: frames; memory: peak GiB; runtime: encoder/decoder seconds.

Method	T	Params	GFLOPs ↓	Mem. ↓	MSE ↓	MAE ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Enc./Dec.
Wan2.1	32	126.9M	14,685.8	1.669	4.65×10^{-4}	0.0108	35.72	0.958	**0.0182**	0.2005/0.3423
CogVideoX	32	206M	33,014.4	4.924	1.12×10^{-3}	0.0148	32.30	0.939	0.0380	0.4403/0.8770
IV-VAE	29	241.9M	33,132.6	2.082	$\mathbf{3.73\times10^{-4}}$	**0.0096**	**36.84**	**0.966**	0.0188	0.9006/0.8279
WF-VAE	33	316M	12,953.2	2.693	4.42×10^{-4}	0.0100	36.29	0.962	0.0227	0.0700/0.2387
LeanVAE	29	40M	328.3	**0.251**	5.01×10^{-4}	0.0111	35.34	0.955	0.0200	0.0114/0.0210
OctVideo	32	**38.2M**	**189.4**	0.803	4.83×10^{-4}	0.0106	36.12	0.960	0.0220	**0.0105/0.0091**

Threshold-controlled inference. Lowering the construction thresholds increases the token budget and improves PSNR and LPIPS, with diminishing gains over the evaluated range (Fig. 4). This provides a quality–token trade-off without retraining the VAE.

4.2.1 Zero-Shot DAVIS Reconstruction

We next test zero-shot high-resolution transfer on all 50 sequences of DAVIS 2016 at their original resolution (Perazzi et al., 2016), using frozen K400 checkpoints without DAVIS fine-tuning or model selection. Each method retains its native temporal support under either native or tiled high-resolution inference. The comparison spans industrial and academic VAEs of different scales. Metrics are averaged equally over aligned full-resolution outputs; details are in Section A.2.

Table 2: Zero-shot reconstruction on all 50 DAVIS 2016 sequences at their original resolution. Metrics are computed on complete output frames using method-specific paths: Wan2.1 uses native full-frame inference; HunyuanVideo, LeanVAE, and OctVideo use non-overlapping 256^2 tiles. T records the effective number of input frames. GFLOPs are averaged over the 50 sequences using $1\ \mathrm{MAC} = 1\ \mathrm{FLOP}$. No model is fine-tuned on DAVIS.

Method	T	PSNR ↑	SSIM ↑	LPIPS ↓	rFVD ↓	VAE s/clip ↓	GFLOPs ↓
Wan2.1 (2025)	32	**33.47**	**0.904**	**0.0713**	23.0	> 20	466,892
HunyuanVideo (2024)	29	32.73	0.901	0.079	27.1	> 20	1,320,531
LeanVAE (2025)	29	32.68	0.898	0.0768	**18.3**	0.904	12,668
OctVideo	32	32.76	0.891	0.094	22.1	**0.540**	**7815**

OctVideo achieves substantially lower VAE runtime while preserving global structure and local details, yielding visually competitive reconstructions in Fig. 5.

4.3 VAE Ablation Studies

All variants are trained from scratch on split 1 of the UCF101 action-recognition dataset (Soomro et al., 2012) with the same data, latent size, optimization, and model-selection protocol. These controls examine the contributions of the local descriptor, adaptive cell placement, and neural processing to VAE quality and cost.

Table 3: Controlled VAE ablations on all 3,783 UCF101 split-1 validation clips. Sparse variants share the 16-channel $8 \times 32 \times 32$ latent and training recipe. "Matched" denotes approximately the same computational cost as OctVideo; $1\ \mathrm{MAC} = 1\ \mathrm{FLOP}$.

Variant	Controlled change	Avg. tokens	GFLOPs	PSNR ↑	SSIM ↑	LPIPS ↓
Dense Conv3D (small)	lightweight dense baseline	8,192	239.7	32.30	0.928	0.0605
Dense Conv3D (large)	scale dense capacity	8,192	1,780.5	34.12	0.947	0.0399
RGB-only tokens	remove explicit gradients	3,225	189.4	32.70	0.932	0.0537
Random matched tree	replace error-guided placement	3,225	189.4	33.50	0.941	0.0422
Full decoder expansion	remove selective refinement	3,225	835.9	33.90	0.940	0.0403
w/o latent residual	remove dense latent residual	3,225	189.2	32.20	0.913	0.0535
Latent residual only	remove sparse encoder pathway	-	126.1	31.87	0.897	0.0685
OctVideo	complete model	3,225	**189.4**	**34.18**	**0.948**	**0.0382**

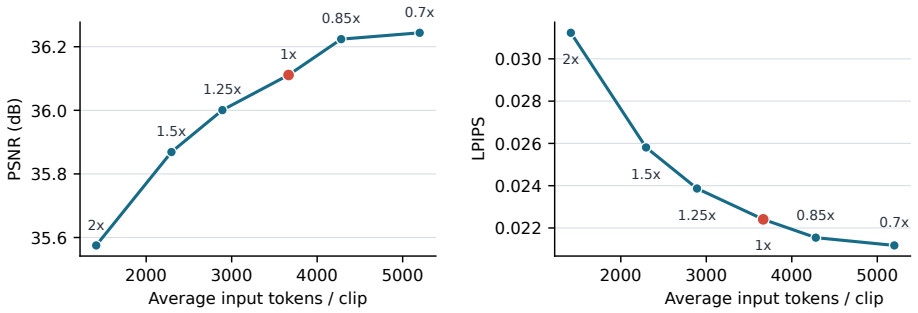

Figure 4: Inference-time threshold sweep on the full Kinetics-400 validation set with a fixed VAE checkpoint. Labels indicate threshold multipliers; red marks the training setting ($1\times$).



Figure 5: Qualitative reconstruction comparison on selected DAVIS 2016 examples.

Adaptive structure improves reconstruction. The small Conv3D baseline is the closest dense alternative in computational cost, yet it trails OctVideo by 1.88 dB and has substantially worse LPIPS. Increasing its capacity recovers nearly the same reconstruction quality (34.12 versus 34.18 dB), but raises computation from 239.7 to 1,780.5 GFLOPs. Among the dense baselines evaluated, increasing capacity approaches OctVideo's reconstruction quality at substantially higher computational cost.

The analytic descriptor matters. Removing the explicit spatio-temporal gradients while preserving the adaptive tree, token width, and network lowers PSNR by 1.48 dB and degrades both SSIM and LPIPS. The gradient channels are therefore useful signal priors rather than redundant inputs that the learned residual can easily replace.

The latent residual complements the sparse pathway. Removing the dense latent residual reduces PSNR by 1.98 dB, while using this branch alone reaches 31.87 dB, 2.31 dB below the complete model. Its much higher LPIPS (0.0685 versus 0.0382) further shows that this dense branch alone cannot match the full model's reconstruction quality. The two pathways therefore provide complementary reconstruction information.

Error-guided allocation improves reconstruction. The random-tree control retains the same RGB+gradient tokens and per-depth counts, but places refinements independently of approximation error; its 0.68 dB drop in PSNR shows that sparsity alone is insufficient. Conversely, fully expanding the decoder raises computation from 189.4 to 835.9 GFLOPs ($4.4\times$) yet yields a PSNR 0.28 dB below that of the selective decoder. Together, these controls support allocating input cells according to local approximation error and using the resulting refinement targets to guide decoder computation.

4.4 VIDEO UNDERSTANDING

As a complementary evaluation beyond reconstruction, we test whether the adaptive representation retains semantic information and supports learned recognition.

Semantic retention under direct reconstruction. A frozen Video Swin model evaluates original videos and their analytic OctVideo reconstructions (Table 4). Top-1 accuracy decreases from 82.27% to 79.51%. Because neither a VAE nor adaptation to the reconstructed inputs is involved, this measures the effect of the representation on the frozen evaluator.

Table 4: Semantic preservation under direct OctVideo reconstruction. A frozen Video Swin model evaluates the original and reconstructed videos.

Input	Top-1	Top-5	ΔTop-1	ΔTop-5
Original video	82.27	96.80	–	–
Direct reconstruction	79.51	94.40	−2.76	−2.40

Recognition learned on OctVideo. Parameter-free sibling pooling reduces the approximately 4,200 input tokens to 3,700 and 800 across classifier stages (Table 6). We train the classifier from scratch on Kinetics-400 without external pre-training. It receives 32 frames at 256×256 resolution and is evaluated with four temporal clips and three spatial crops; token counts below are for one view.

Table 5: Action recognition on Kinetics-400. Token counts indicate the per-view sequence length after initial embedding; OctVideo reports the mean number of adaptive leaves.

Method	Pre-train	Params	Input	Views	Tokens/view	Top-1	Top-5
Rev-MViT-B (Mangalam et al., 2022)	none	34.9M	16×224^2	5×1	25,088	78.5	93.4
MViTv1-B (Fan et al., 2021)	none	36.6M	32×224^2	5×1	50,176	80.2	94.4
MViTv2-S (Li et al., 2022)	none	34.5M	16×224^2	5×1	25,088	81.0	94.6
OctVideo	none	31M	32×256^2	4×3	∼4,200	80.1	94.4
Video Swin-T (Liu et al., 2022)	IN-1K	28.2M	32×224^2	4×3	50,176	78.8	93.6
Hiera-B (Ryali et al., 2023)	K400 MAE	51M	16×224^2	5×3	25,088	84.0	–
VideoMAE-S (Tong et al., 2022)	K400 SSL	22M	16×224^2	5×3	1,568	79.0	93.8

With 31M parameters and approximately 4,200 input tokens, OctVideo reaches 80.1% Top-1 and 94.4% Top-5 accuracy. Transferring the K400 checkpoint to UCF101 split 1 yields 96.43% and 99.52%, respectively; details are in Table 7.

5 CONCLUSION

We presented OctVideo, an adaptive spatio-temporal representation in which local signal approximation determines cell refinement. RGB values and gradients describe the video within each cell, preserving coarse descriptions of smooth regions alongside finer ones for detailed regions. Learned residual features supplement these descriptors for neural processing. Sequence-based Conv1D encoding, alignment to a regular latent grid, and selective refinement make the representation practical for video reconstruction. Experiments on Kinetics-400 and zero-shot transfer to DAVIS demonstrate competitive pixel-level fidelity at low computational cost. Controlled UCF101 ablations support the design choices: naive dense scaling is costly, explicit gradients improve the token signal, and error-guided input and decoder topologies outperform unselective alternatives. Complementary recognition experiments show that the same adaptive cells retain semantic information and support learned classification through patch-wise attention and sibling pooling.

These results show that a compact signal description can also provide useful structure for neural computation. Improving fine spatio-temporal detail and exploring the benefits of fast decoding in repeated video sampling and evaluation are promising directions for future work.

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.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your ViT but faster. In *International Conference on Learning Representations*, 2023.

Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6299–6308, 2017.

Yu Cheng and Fajie Yuan. Leanvae: An ultra-efficient reconstruction vae for video diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15692–15702, 2025.

Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6824–6835, 2021.

Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6202–6211, 2019.

Sarah F. Frisken, Ronald N. Perry, Alyn P. Rockwood, and Thouis R. Jones. Adaptively sampled distance fields: A general representation of shape for computer graphics. In *SIGGRAPH*, 2000.

Xinyue Hao, Gen Li, Shreyank N. Gowda, Robert B. Fisher, Jonathan Huang, Anurag Arnab, and Laura Sevilla-Lara. Principles of visual tokens for efficient video understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 21254–21264, 2025.

Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, et al. Hunyuanvideo: A systematic framework for large video generative models, 2024.

Yeonkyeong Lee, Hyunsung Go, Jongmin Kim, Sewoong Lim, and Donghoon Lee. Keep-or-drop? adaptive tokenizer for compact video representation, 2026.

Yanghao Li, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Mvitv2: Improved multiscale vision transformers for classification and detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4804–4814, 2022.

Zongjian Li, Bin Lin, Yang Ye, Liuhan Chen, Xinhua Cheng, Shenghai Yuan, and Li Yuan. Wf-vae: Enhancing video vae by wavelet-driven energy flow for latent video diffusion model. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 17778–17788, 2025.

Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7083–7093, 2019.

Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3202–3211, 2022.

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.

Karttikeya Mangalam, Haoqi Fan, Yanghao Li, Chao-Yuan Wu, Bo Xiong, Christoph Feichtenhofer, and Jitendra Malik. Reversible vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10830–10840, 2022.

Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. Multi-level partition of unity implicits. *ACM Trans. Graph. (SIGGRAPH)*, 22(3), 2003.

Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 724–732, 2016.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. DynamicViT: Efficient vision transformers with dynamic token sparsification. In *Advances in Neural Information Processing Systems*, 2021.

Tomer Ronen, Omer Levy, and Avram Golbert. Vision transformers with mixed-resolution tokenization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 4613–4622, 2023.

Chaitanya Ryali, Yuan-Ting Hu, Daniel Bolya, Chen Wei, Haoqi Fan, Po-Yao Huang, Vaibhav Aggarwal, Arkabandhu Chowdhury, Omid Poursaeed, Judy Hoffman, Jitendra Malik, Yanghao Li, and Christoph Feichtenhofer. Hiera: A hierarchical vision transformer without the bells-and-whistles. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 29441–29454, 2023.

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.

Onkar Susladkar, Tushar Prakash, Adheesh Juvekar, Kiet A. Nguyen, Dong-Hwan Jang, Inderjit S. Dhillon, and Ismini Lourentzou. PyraTok: Language-aligned pyramidal tokenizer for video understanding and generation, 2026.

Feilong Tang, Xiang An, Yunyao Yan, Yin Xie, Bin Qin, Kaicheng Yang, Yifei Shen, Yuanhan Zhang, Chunyuan Li, Shikun Feng, et al. Onevision-encoder: Codec-aligned sparsity as a foundational principle for multimodal intelligence. *arXiv preprint arXiv:2602.08683*, 2026.

Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. VideoMAE: Masked autoencoders are data-efficient learners for self-supervised video pre-training. In *Advances in Neural Information Processing Systems*, volume 35, pp. 10078–10093, 2022.

Du Tran, Lorenzo Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4489–4497, 2015.

Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6450–6459, 2018.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *NeurIPS*, 2017.

Wan-Video. Wan: Open and advanced large-scale video generative models, 2025.

Limin Wang, Bingkun Huang, Zhiyu Zhao, Zhan Tong, Yinan He, Yi Wang, Yali Wang, and Yu Qiao. VideoMAE V2: Scaling video masked autoencoders with dual masking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14549–14560, 2023.

Peng-Shuai Wang. OctFormer: Octree-based transformers for 3D point clouds. *ACM Trans. Graph. (SIGGRAPH)*, 42(4), 2023.

Peng-Shuai Wang, Chun-Yu Sun, Yang Liu, and Xin Tong. Adaptive O-CNN: A patch-based deep representation of 3D shapes. *ACM Trans. Graph. (SIGGRAPH ASIA)*, 37(6), 2018.

Peng-Shuai Wang, Yang Liu, and Xin Tong. Dual octree graph networks for learning adaptive volumetric shape representations. *ACM Trans. Graph. (SIGGRAPH)*, 41(4), 2022.

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.

Pingyu Wu, Kai Zhu, Yu Liu, Liming Zhao, Wei Zhai, Yang Cao, and Zheng-Jun Zha. Improved video vae for latent video diffusion model. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 18124–18133, 2025.

Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. VideoGPT: Video generation using VQ-VAE and transformers, 2021.

Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. In *International Conference on Learning Representations*, volume 2025, pp. 83048–83077, 2025.

Sijie Zhao, Yong Zhang, Xiaodong Cun, Shaoshu Yang, Muyao Niu, Xiaoyu Li, Wenbo Hu, and Ying Shan. Cv-vae: A compatible video vae for latent generative video models, 2024.

Yian Zhao, Feng Wang, Qiushan Guo, Chang Liu, Xiangyang Ji, Jian Zhang, and Jie Chen. Video generation with predictive latents, 2026.

A Appendix

A.1 Video Understanding Details

Table 6 reports how the adaptive input is reduced across the recognition hierarchy. Pooling is parameter-free: complete groups of eight children are replaced by their parent, while shallower leaves pass through.

Table 6: K400 recognition token hierarchy for one $32 \times 256 \times 256$ view sampled at temporal interval 2. Rows after the sparse input are classifier states produced by sibling pooling.

Level	Active depths	Operation	Consumer	Avg. tokens
Sparse input	d_3–d_6	adaptive leaves	classifier	∼4,200
Level-5 state	d_3–d_5	$d_6 \to d_5$ pool	classifier	∼3,700
Level-4 state	d_3–d_4	$d_5 \to d_4$ pool	classifier	∼800

UCF101 transfer. Starting from the K400-supervised OctVideo checkpoint, we replace the classification head and fine-tune on the official UCF101 split 1 (Soomro et al., 2012). No UCF101 video is used in pre-training. Fine-tuning uses AdamW for 50 epochs with a base learning rate of 3×10^{-5} and weight decay of 0.05.

Table 7: UCF101 transfer and representative RGB-based methods. Published methods retain their original protocols; OctVideo uses four temporal clips and three spatial crops on official split 1.

Method	Pretraining	Top-1	Top-5
C3D (3 nets) (Tran et al., 2015)	I380K+Sports-1M	85.2	–
I3D-RGB (Carreira & Zisserman, 2017)	ImageNet+K400	95.6	–
R(2+1)D-RGB (Tran et al., 2018)	K400	96.8	–
TSM (Lin et al., 2019)	K400	95.9	99.7
VideoMAE V2-g (Wang et al., 2023)	UnlabeledHybrid+K710	**99.6**	**100.0**
OctVideo (ours)	K400	96.43	99.52

A.2 Training and Evaluation Protocols

Optimization. The K400 classifier uses AdamW with $\beta = (0.9, 0.999)$ and weight decay 0.05. It is trained for 200 epochs at a base learning rate of 10^{-4}, with 30 epochs of linear warm-up from $0.01\times$ the base learning rate, followed by cosine decay to 10^{-6}. The per-GPU batch size is two with two-step gradient accumulation. Gradients are clipped to norm 1.0, stochastic depth increases linearly from 0 to 0.2, and an exponential moving average with rate 0.999 is maintained. The VAE uses MuonAdamW: Muon updates large matrix parameters and AdamW updates the remaining parameters. It is trained for 50 epochs at 6×10^{-4}, with a two-epoch warm-up from $0.1\times$ the base learning rate and cosine decay to 10^{-6}, using one clip per GPU and eight-step gradient accumulation. Muon uses momentum 0.95, five Newton–Schulz steps, Nesterov momentum, and weight decay 0.05; AdamW uses $\beta = (0.9, 0.999)$ and $\epsilon = 10^{-8}$. Both models use BF16 automatic mixed precision. The VAE maintains an exponential moving average with rate 0.999 and is trained without an adversarial loss.

Classification augmentation. We follow the MViT augmentation recipe (Fan et al., 2021; Li et al., 2022), including random spatial crops and horizontal flips, RandAugment, Mixup, CutMix, and random erasing. Specifically, we use two augmented views per training video, four RandAugment operations of magnitude 7, color jitter with probability 0.8 and magnitude 0.4, random erasing with probability 0.25, Mixup $\alpha = 0.8$, CutMix $\alpha = 1.0$, and label smoothing 0.1. Mixup or CutMix is applied with probability 1.0, with equal probability of selecting either operation.

High-resolution reconstruction. For the DAVIS comparison in Table 2, we extract one center-aligned clip of each model's native length from each of the 50 sequences and use frozen checkpoints. Wan2.1 uses native full-frame inference (32 frames, internally padded to 33, with four-GPU layer-wise parallelism). HunyuanVideo, LeanVAE (29 frames, FP16), and OctVideo use non-overlapping

256×256 tiles. Inputs are replicate-padded when needed, and outputs are cropped after full-frame reconstruction or tile stitching.

Metrics are computed on all valid aligned frames after restoring the original resolution and are averaged equally across sequences. PSNR and SSIM use RGB values in $[0, 1]$, LPIPS uses the AlexNet backbone for every method, and rFVD uses one Kinetics-400 I3D feature per aligned clip.

A.3 Network Architectures

VAE architecture. Figure 6 summarizes the encoder and the two residual branches. Table 8 gives the complete architecture used in our main experiments. Every leaf stores a $2 \times 4 \times 4$ grid of 12-dimensional RGB-and-gradient descriptors, flattened into one 384-dimensional token. A depth-d_c leaf covers $(2s, 4s, 4s)$ video samples, with $s = 2^{D-d_c}$; only at the finest depth D is its coverage $2 \times 4 \times 4$.

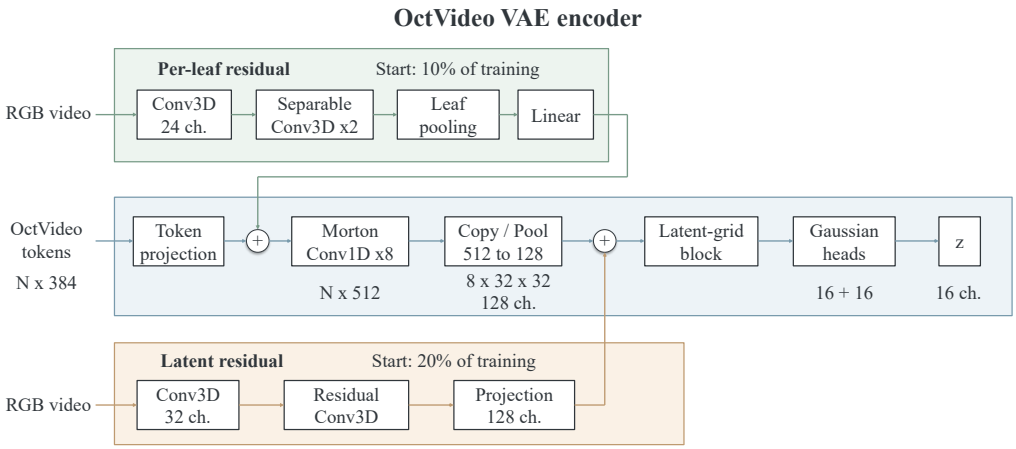

Figure 6: OctVideo VAE encoder. The per-leaf branch pools finest-level features according to the OctVideo partition and adds them before sequence encoding. The latent residual is added after copy/pool mapping and before the latent-grid block. The branches are enabled after 10% and 20% of the total training epochs, respectively. Gaussian heads predict the mean and log variance; z is sampled during training and uses the mean at evaluation. Normalization, activations, and positional embeddings are omitted for clarity.

The per-leaf residual branch first applies a 3D convolution from 3 to 24 channels with kernel and stride $(2, 4, 4)$, followed by SiLU. Two separable 3D layers each use a 3^3 depthwise convolution and a 1^3 pointwise convolution, with SiLU after both convolutions. We recursively construct a feature pyramid by 2^3 pooling and concatenate the mean and maximum over each leaf support. A zero-initialized linear layer maps the resulting 48 features to 512 channels. Its learnable scalar is initialized to one, and the branch is activated after 10% of the total training epochs.

The analytic token is processed by LayerNorm, a $384 \to 512$ linear layer, and GELU; the per-leaf residual and a learned depth embedding are then added. Eight pre-normalized Morton Conv1D blocks use width 512, depthwise kernel 7, dilations $(1, 2, 4, 1, 2, 4, 1, 1)$, and an $512 \to 2048 \to 512$ GELU MLP. Copy/pool resampling first projects features from 512 to 128 channels. Coarse leaves are copied to every covered latent location, whereas each complete group of eight finest-depth siblings is averaged into one location, producing an $8 \times 32 \times 32 \times 128$ grid.

The dense latent residual applies a 3D convolution from 3 to 32 channels with kernel and stride $(4, 8, 8)$ and SiLU. It then uses one residual block composed of GroupNorm, a 3^3 depthwise convolution, SiLU, and a 1^3 pointwise convolution. A zero-initialized 1^3 convolution projects to 128 channels; its scale is initialized to one and the branch is activated after 20% of the total training epochs. Its output is added to the copy/pool grid before the latent-grid block. The latter uses GroupNorm, a 3^3 depthwise convolution, and a $128 \to 128 \to 128$ pointwise GELU MLP, all with residual connections. After LayerNorm, separate linear heads predict 16 means and 16 log variances per grid location.

Table 8: Layer-by-layer specification of the OctVideo VAE. Shapes omit the batch dimension. DW denotes a depthwise convolution.

Component	Input → output	Layers and hyperparameters	Activation / normalization
Analytic token projection	$N \times 384 \to N \times 512$	Linear $384 \to 512$	LayerNorm, GELU
Per-leaf residual	$3\times32\times256\times256 \to N \times 512$	Conv3D $(2, 4, 4)$, 24 ch.; 2×[DWConv3D 3^3, Conv3D 1^3]; mean+max pool; Linear $48 \to 512$	SiLU; zero-init. output; start at 10%
Morton sequence encoder	$N \times 512 \to N \times 512$	8 blocks; DWConv1D $k = 7$; dilations $1, 2, 4, 1, 2, 4, 1, 1$; MLP ratio 4	Pre-LayerNorm, GELU
Copy/pool mapping	$N \times 512 \to 8 \times 32 \times 32 \times 128$	Linear $512 \to 128$; copy coarse leaves / mean-pool 8 children	learned 3D position embedding
Dense latent residual	$3\times32\times256\times256 \to 8 \times 32 \times 32 \times 128$	Conv3D $(4, 8, 8)$, 32 ch.; 1 residual [DWConv3D 3^3, Conv3D 1^3]; Conv3D 1^3 to 128	GroupNorm, SiLU; zero-init. output; start at 20%
Latent-grid block	$8 \times 32 \times 32 \times 128$	1 block; DWConv3D 3^3; pointwise MLP ratio 1	GroupNorm, GELU
Posterior heads	$8192 \times 128 \to 8192 \times (16 + 16)$	separate Linear $128 \to 16$ for $\boldsymbol{\mu}$ and $\log \boldsymbol{\sigma}^2$	LayerNorm
Latent decoder stem	$8192\times16 \to 8\times32\times32\times384$	Linear $16 \to 384$; learned 3D position embedding	–
Coarse decoder	$8 \times 32 \times 32 \times 384$	3 residual blocks; Conv3D 3^3; MLP $384 \to 768 \to 384$	LayerNorm, SiLU/GELU
Split and expansion	coarse grid → selected $16 \times 64 \times 64$ cells	LN–Linear $384 \to 1$; Linear $384 \to 8 \times 384$; 8 octant embeddings	sigmoid threshold 0.5
Selective refinement	selected cells, 384 ch.	2 Morton DWConv1D blocks, $k = 7$, dilations $(1, 2)$; MLP ratio 2	Pre-LayerNorm, GELU
Canvas fusion	coarse/refined cells → $24 \times 32 \times 128 \times 128$	Linear $384 \to 24 \cdot 4^3$ (coarse) or $384 \to 24 \cdot 2^3$ (refined)	coordinate placement
RGB head	$24 \times 32 \times 128 \times 128 \to 3\times32\times256\times256$	DWConv3D 3^3; Conv3D 1^3, $24 \to 32$; nearest spatial 2×; DWConv3D 3^3; Conv3D 3^3, $32 \to 3$	SiLU

VAE objective and optimization. The reconstruction term is RGB ℓ_1. LPIPS uses the AlexNet backbone and 16 randomly sampled frames per clip, with weight 0.5. The KL weight is linearly increased from zero to 10^{-6} over the first 10,000 optimizer steps. Split logits use binary cross-entropy with weight 0.05; the positive-class weight is estimated per batch and capped at 64. During training, ground-truth split masks determine expansion; inference uses a sigmoid threshold of 0.5. Optimization begins with the octree pathway alone. The per-leaf and latent residual branches are activated after 10% and 20% of the total training epochs, respectively, and are then trained jointly with all other VAE parameters.

Classification network. The classifier receives the same 384-dimensional analytic OctVideo tokens; the dense residual branch is disabled for this experiment. A linear layer maps tokens to width 512, and a two-layer $1 \to 512 \to 512$ SiLU MLP embeds normalized octree depth. The ten pre-LayerNorm transformer blocks are divided into stages of $(4, 4, 2)$ blocks. Each block uses 8 attention heads (64 channels per head), 3D rotary position encoding, QKV bias, attention over contiguous groups of 256 hierarchy-ordered tokens, and a $512 \to 2048 \to 512$ GELU MLP. Attention and projection dropout are zero; stochastic depth grows linearly from 0 to 0.2 across the ten blocks. After stages one and two, complete groups of eight depth-6 and depth-5 siblings, respectively, are averaged into their parents; unmatched shallower leaves pass through unchanged. A final FP32 LayerNorm is followed by a learned single-query attention readout with independent 512-dimensional query, key, and value projections, and a $512 \to 400$ classifier.

Recognition and reconstruction use the same thresholds and tree-construction rule, but not the same temporal sampling protocol. Recognition samples 32 frames at interval 2 and averages approximately

Table 9: OctVideo classification architecture on K400.

Stage	Blocks	Token operation	Feature configuration
Input	–	Linear $384 \to 512$ + depth embedding	width 512
Stage 1	4	patch attention, then $d_6 \to d_5$ pool	8 heads, chunk 256, MLP $4\times$
Stage 2	4	patch attention, then $d_5 \to d_4$ pool	8 heads, chunk 256, MLP $4\times$
Stage 3	2	patch attention	8 heads, chunk 256, MLP $4\times$
Readout	–	single-query attention + Linear $512 \to 400$	final FP32 LayerNorm

4,200 input leaves, whereas the VAE uses 32 consecutive frames (interval 1) and averages 3,665 leaves on K400 validation. The larger recognition count therefore reflects the greater temporal span and variation of its input, rather than data augmentation or a different tree threshold.

A.4 Tree Construction and Direct Reconstruction

Tree construction. Starting from the depth-3 partition, we fit RGB values and gradients within each of a candidate cell's 32 subregions using Eq. (2), then evaluate its maximum approximation error. A cell is retained as a leaf if its error does not exceed the threshold; otherwise, it is split into eight children, which are tested recursively. The thresholds at depths 3, 4, and 5 are 0.7, 0.8, and 1.0, respectively; depth-6 cells are retained without further subdivision. The same schedule is used for reconstruction and recognition. The leaves cover the full video without overlap, and their geometric centers determine the Morton ordering used for subsequent sequence processing. Figure 7 shows how scaling these thresholds trades the number of leaves for direct reconstruction quality.

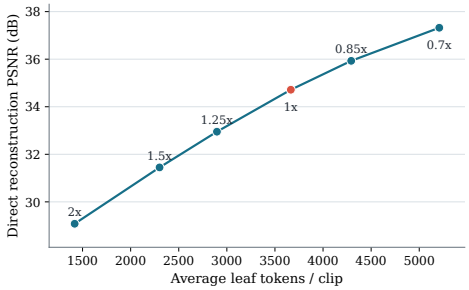

Figure 7: Direct reconstruction as all three construction thresholds are scaled together on K400 validation. Smaller multipliers create more leaves and improve analytic reconstruction. The red point is our default schedule $(0.7, 0.8, 1.0)$, which averages 3,665 leaves and 34.72 dB. PSNR is computed per clip and then averaged.

Efficient implementation. We construct the tree level by level on the GPU, fitting and testing active cells in parallel with batched tensor operations. Only cells that fail the error test generate candidates at the next level; accepted leaves are stored in flat tensors, avoiding per-cell Python recursion and pointer-based traversal.

Direct reconstruction. At each video coordinate, we find the covering leaf c and its subregion q, then evaluate $\hat{V}_{c,q}$. Combining these local predictions recovers the complete video without a VAE or learned residual features. With PSNR computed per clip and then averaged, this analytic reconstruction achieves 34.72 dB on K400.

Figure 8 compares analytic reconstructions with controlled variations in cell content and placement.

These controls show that OctVideo is more than a reduced token budget. With the adaptive tree fixed, removing the gradient channels introduces visible block artifacts and weakens thin contours. Matching the number of cells at every depth but placing them randomly instead concentrates severe errors on foreground structure. In contrast, the complete representation allocates fine cells to the dancer while keeping the smooth ground coarse. This comparison shows how the local descriptor and adaptive cell placement preserve visual structure within a compact representation.



Figure 8: Direct reconstruction of a Kinetics-400 frame without a learned VAE. We compare the original frame with full OctVideo, an RGB-only variant on the same adaptive tree, and a random tree matched by per-depth token counts. Explicit gradients and error-guided cell placement better preserve object boundaries and fine structure.

A.5 KINETICS-400 VAE RECONSTRUCTIONS

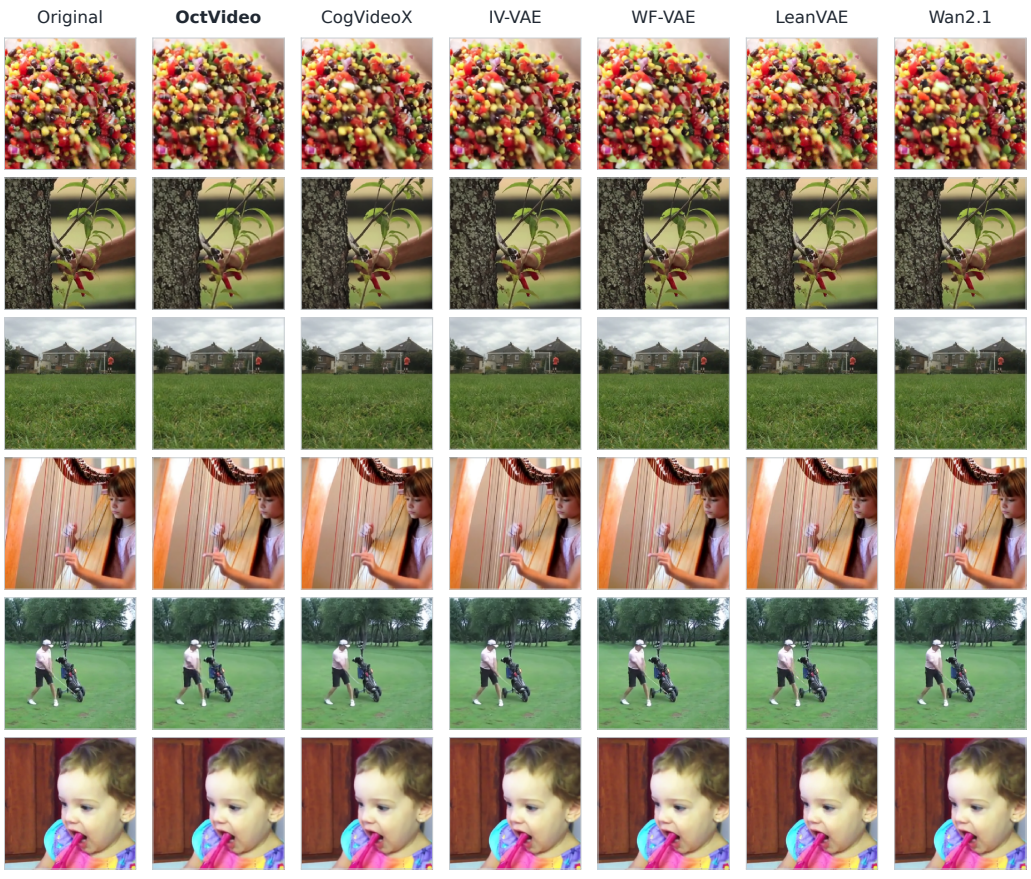

Figure 9: Qualitative VAE reconstructions on Kinetics-400. Each row shows the same source frame across the original clip and six reconstructions. WF-VAE uses 33 input frames; OctVideo, CogVideoX, and Wan2.1 use 32; IV-VAE and LeanVAE use 29.

AI USE STATEMENT

The core research idea, experimental design, and reported measurements are the authors' own work. Generative AI tools assisted with code organization, discovery of relevant literature, manuscript editing, and suggestions on presentation and research directions. The authors reviewed AI-assisted material and take full responsibility for the methods, results, and claims.