arXiv:2610.33857v1 [cs.CV] 24 Oct 2026

TrackTok: Object-Centric Video Tokenization with Semantically Persistent Tokens

Nhi Pham[1]* **Christopher Wewer**[1]* **Bernt Schiele**[1] **Jonas Fischer**[1]† **Jan Eric Lenssen**[1,2]†

[1]Max Planck Institute for Informatics, Saarland Informatics Campus, Germany
[2]Saarland University, Saarland Informatics Campus, Germany
{nhipham, cwewer, schiele, jofische, jlenssen}@mpi-inf.mpg.de

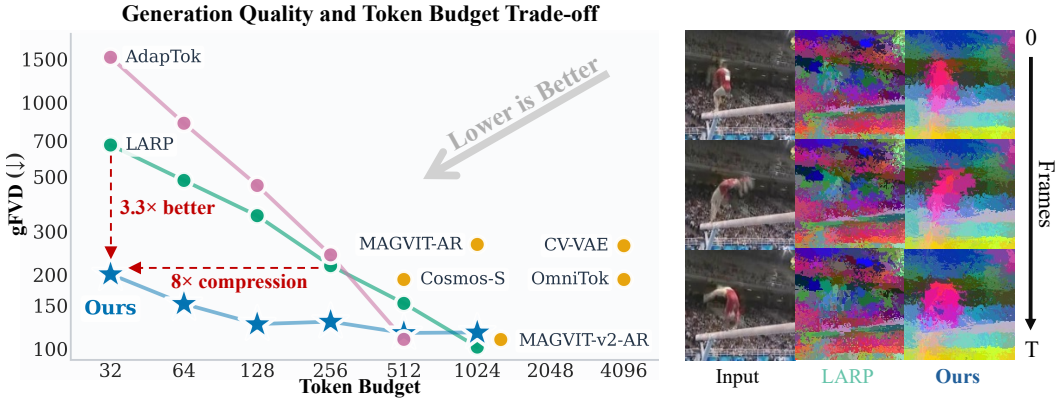


Figure 1: **TrackTok efficiently encodes videos in object-aligned tokens.** We introduce TrackTok, a novel 1D video tokenizer that enables high compression with object-aligned, semantically persistent tokens. **Left:** TrackTok provides a new frontier for efficient video tokenizers, yielding $8\times$ smaller token sets at comparable class-conditional gFVD on UCF-101 dataset compared to state-of-the-art LARP [1] and AdapTok [2]. **Right:** With the same 128-token budget, LARP's tokens are locked to fixed spatial locations, while TrackTok's tokens dynamically follow semantic units across frames.

Abstract

We introduce TrackTok, an efficient video tokenization framework that produces object-centric tokens, i.e., tokens that consistently encode the same semantic objects across time, even as it moves, deforms, or undergoes partial occlusion. Conventional patch-based or existing holistic tokenizers produce token identities often tied to fixed image locations. In contrast, our approach learns temporally persistent object-centric representations that bind visual features across frames into stable semantic units. This enables substantially more compact, continuous latent video representations, reducing the number of tokens required for downstream generative modeling with state-of-the-art flow models, while preserving the semantic structure needed for high-quality synthesis. Our tokenizer offers a favorable quality-efficiency trade-off. TrackTok achieves comparable class-conditional generative FVD on UCF-101 dataset at $8\times$ fewer latent tokens and better latency, and outperforms existing 1D tokenization baselines on frame prediction task on Kinetics-600 across low token budgets. These results suggest that object-aligned and temporally persistent tokenization is a promising direction for scalable video generation, where reducing latent token counts is critical for efficient modeling. Our code for this work is available at github.com/phamleyennhi/tracktok.

*Equal contribution.
†Equal advising.

1 Introduction

Autoencoding videos into compressed latent spaces is an active research area with large impact on efficient video streaming and generative modeling [3, 4, 5]. An optimal autoencoder should encode videos into a representation that is (1) highly compressive and (2) allows for high-quality video generation in latent space, i.e., being easy to process by current-gen, large-scale generative models, such as video diffusion models [6, 7, 8] or autoregressive transformers [9, 10, 11]. We refer to *compression* as the number of latent tokens a video is encoded into, as it is the quantity that primarily governs the cost of the downstream generative model: attention in diffusion transformers over continuous latents scales in the token count, while for autoregressive transformers over discrete latents the token count directly equals the number of network function evaluations (NFEs). A promising recent direction are tokenizers that map videos into unstructured sets of tokens, which are not aligned with the spatiotemporal grid of input video tokens [1, 12]. In this paper, we present `TrackTok`, a novel tokenizer of this category that represents videos as semantic, object-centric tokens, achieving a state-of-the-art trade-off in compression vs. generation quality (c.f. Fig. 1 left).

Videos often contain redundant information: a set of entities is moving through an otherwise static scene, where the existence of an individual entity remains persistent while its location is changing. Current tokenizers do not fully leverage this redundancy as they compress video patches but maintain the spatial and temporal dimensions [13, 14, 15]. We show that even existing holistic autoencoders [1], while already compressing videos into 1D tokens sets or sequences, still fall back to encoding fixed spatial locations into individual tokens. Thus, when an object moves through a scene and changes location in pixel space, the representing token changes multiple times across the video, failing to disentangle the object's existence and its location, and thus failing to utilize the given redundancy.

In contrast, `TrackTok` clusters the scene into individual semantic regions, usually objects, in an unsupervised manner and represents them persistently with the same tokens over the course of the video (c.f. Fig. 1 right). We achieve this by designing a perceiver-based encoder that leverages V-JEPA 2.1 [16] as feature extractor and grounds the cross-attention mechanism between latent and input video tokens in semantic, *input-dependent* cluster queries instead of *input-independent* positional embeddings. Thus, latent token identity is defined by semantic features instead of arbitrary optimized embeddings. We find that this mechanism provides a drastically different inductive bias, leading to latent tokens specializing on semantic instances instead of spatiotemporal locations.

On YouTube-VIS dataset [17] with the same token budget, TrackTok's latent tokens are more temporally consistent and more strongly aligned with individual objects than those of 1D holistic tokenizer LARP [1]. Furthermore, we show that `TrackTok` presents a new frontier for the compression vs. video generation quality trade-off, achieving strong class-conditional generation and frame-prediction quality (gFVD), with as low as $8\times$ **fewer tokens** and **better latency** compared to direct baselines.

In summary, our contributions are:

- We introduce `TrackTok`, a novel video tokenizer that encodes videos into a 1D set of object-aligned, latent tokens.
- We present a novel mechanism to introduce inductive biases into perceiver architectures, controlling which property is used by the network to cluster via cross-attention. By using semantic cluster queries, the model learns to perform attention based on object semantics.
- We show that `TrackTok` improves on the state-of-the-art trade-off between compression and latent video generation quality.

2 Related Work

State-of-the-art image [13, 18] and video [6, 7, 8] generation pipelines mostly follow a two-stage paradigm consisting of a tokenizer and a latent generator. Recent works explore the tokenizer design space for stronger compression and the integration of vision foundation models for better training efficiency (Sec. 2.1). We further discuss `TrackTok`'s relation to object-centric learning in Sec. 2.2.

2.1 Vision Tokenizers for Generation

In the context of visual generation, tokenizers have been initially proposed for discretization of continuous images into tokens for autorgressive transformers [19]. With the success of latent

diffusion and flow models [13], the goal of continuous tokenization has shifted towards compression for more efficient latent generation.

Spatial Tokenization. An intuitive design choice for vision tokenizers are convolutional autoencoders [13, 19, 14, 15] that progressively downsample images or videos into lower-resolution feature maps or grids. VidTok [20] decouples spatial and temporal sampling, achieving strong reconstruction fidelity. This paradigm has been successfully implemented in large-scale text-to-image [13, 18] and -video [6, 7, 8] models, offering direct compatibility with spatial conditioning [21] or autoregressive long video generation [22]. However, it misses potential for stronger compression, as it maintains the spatial 2D/3D structure, rendering such models ineffective for high-throughput video tasks.

Adaptive Tokenization. Another line of work makes the token budget input-dependent. TokenLearner [23] learns spatial attention maps over intermediate features to obtain adaptive tokens. ElasticTok [24] encodes video blocks into a variable token count, dropping a random tail of each block's token sequence in training. As noted by [2], its latents remain tied to local spatiotemporal regions. AdapTok [2] combines tail-drop with a block-causal 1D latent space and a learned scorer, distributing a global token budget across blocks and samples, and reports state-of-the-art generation among causal 1D video tokenizers. We compare against AdapTok as the strongest adaptive baseline.

1D Fixed-length Tokenization. Recent work [25, 26, 27, 1, 12, 28] proposes 1D token sets as latent representations. TiTok [25] compresses images into a fixed-size set of just 32 tokens using transformer architecture. Our main baseline LARP [1] extends this paradigm to videos, requiring $> 1k$ tokens to achieve sufficient reconstruction quality. FlexTok [28] resamples images into flexible-length 1D token sequences acting as conditioning for a generative decoder. All these methods are optimized for autoregressive generation with discrete tokens from a fixed-size codebook. This limits compression in comparison with continuous tokenizers, as only the choice of tokens but not the tokens themselves are input-dependent. We propose `TrackTok`, a 1D *continuous* tokenizer for latent video diffusion, achieving stronger compression by encoding vision foundation features into object-centric tokens.

Foundation Models for Visual Tokenization. Following success in leveraging vision foundation models (VFMs) like DINO [29, 30] to speed up training of latent diffusion models [31], recent works investigate the use of VFMs for continuous tokenizers. VA-VAE [32] aligns latents of a 2D VAE with DINO features, boosting training efficiency. RAE [33] and VFM-VAE [34] directly use DINO as frozen encoder and train image decoders on top. However, these approaches are limited to generation with latent tokens representing patch-level information. In contrast, with a much smaller token set, `TrackTok` achieve similar video generation quality by learning *object-centric* tokens. This object-centricity is key to simultaneously enable strong compression and latent diffusability. Although existing approaches above learn 1D token sequences, they lack this property, showing strong spatial biases of tokens. Our representation is especially useful for videos, as the tokens are encouranged to be *semantically persistent over time*.

2.2 Object-Centric Representation Learning

Object-centric learning (OCL) aims to represent images or videos as latent variables that correspond to individual physical objects. Slot Attention [35] popularized OCL by introducing a modular bottleneck in form of a permutation-invariant set of slots that learn to bind to objects, trained end-to-end for reconstruction. SAVi [36, 37], STEVE [38], and SlotContrast [39] extend this paradigm to videos. With the rise of VFMs, recent works [40, 41, 42] combine slot attention with semantic feature extraction and self-supervision objectives. These object-centric autoencoders are not designed as video tokenizers for latent video generation. They are optimized for unsupervised object discovery and are known for insufficient reconstruction quality due to strong architectural bottlenecks.

To our knowledge, `TrackTok` is the first video tokenizer for latent generation, revisiting ideas of OCL to learn a token representation that is highly compressed, semantically meaningful, and therefore efficient for generative modeling with state-of-the-art diffusion and flow models.

3 `TrackTok`

We present a novel video autoencoder architecture, `TrackTok`, consisting of a frozen vision foundation model that extracts semantic features from a video, a trained Vision Transformer that encodes semantic feature patches into an object-aligned latent representation, and a trained decoder that reconstructs RGB videos from the encoded semantic information (see Fig. 2).

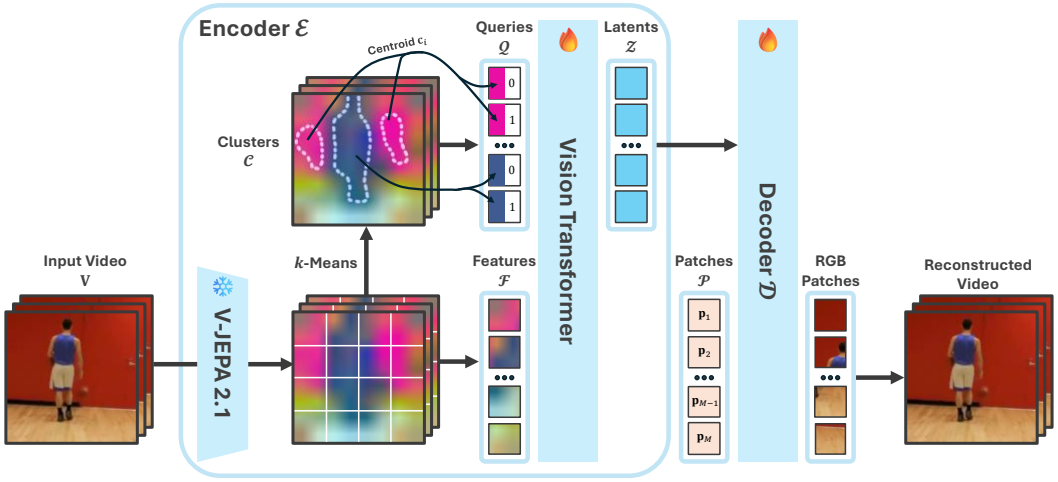


Figure 2: **Method Overview.** `TrackTok` encodes an input video $\mathbf{V}$ into a 1D set of object-centric, semantically persistent, latent tokens $\mathcal{Z}$. Its encoder $\mathcal{E}$ is composed of semantic feature extraction with V-JEPA 2.1 [16], the construction of input-dependent queries $\mathcal{Q}$ using centroids of k-Means clusters $\mathcal{C}$, and a ViT performing attention based on semantic similarity instead of spatial position. The decoder $\mathcal{D}$ reconstructs the three-dimensional RGB video from the 1D latent set $\mathcal{Z}$.

Formally, the encoder $\mathcal{E}$ encodes a video $\mathbf{V} \in [0,1]^{3\times T\times H\times W}$ into a fixed-size set of latent tokens $\mathcal{Z} = \{\mathbf{z}_i \in \mathbb{R}^D\}_{i=1}^N$. We leverage V-JEPA 2.1 [16] as feature extractor and introduce cluster queries to bias the encoder towards learning object-centric, semantically persistent tokens. This representation enables both high compression and efficient training of a latent flow model. Given a set of encoded or generated latent tokens, the decoder $\mathcal{D}$ reconstructs an RGB video. We introduce the encoder design in Sec. 3.1, the decoder in Sec. 3.2, and latent generation in the `TrackTok` space in Sec. 3.3.

3.1 Encoding Videos Into Object-Centric Tokens

Our goal is to learn an object-centric latent representation that encodes and tracks semantically persistent units into individual tokens shared across all frames of a video clip. The key to achieve this goal is to design the right inductive biases for a learnable encoder. Our `TrackTok` encoder consists of three components: **(1) semantic feature extraction** using a pretrained VFM, **(2) our cluster queries** , and **(3) a transformer encoder** that transforms these queries into a compact set of video latents.

Semantic Feature Extraction. We first leverage V-JEPA 2.1 [16] to extract semantic feature patches $\mathcal{F} = \{\mathbf{f}_i \in \mathbb{R}^C\}_{i=1}^F$ from our input video where $F = \frac{H}{16} \cdot \frac{W}{16} \cdot \frac{T}{2}$, as self-supervised VFMs [30] have been shown to learn representations that are implicitly object-aligned and facilitate latent generative modeling [32]. However, directly training a ViT-based encoder on such features does not yield object-centric but patch-wise latent tokens (c.f. Fig. 6), and naively reducing the frozen feature grid to a smaller token count substantially degrades generation quality (c.f. App. D).

Cluster Queries (CQ). Instead, we propose to apply k-Means clustering on the extracted features to obtain K clusters $\mathcal{C}_j$ that correspond to groups of semantically similar features. To obtain a positional encoding for each semantic unit, we first compute frame-wise average cluster positions $\text{pos}(\mathcal{C}_j^t) \in \mathbb{R}^2$

$$\text{pos}(\mathcal{C}_j^t) = \frac{1}{\max\left(1, |\mathcal{C}_j^t|\right)} \sum_{\mathbf{f}\in\mathcal{C}_j^t} \text{pos}(\mathbf{f}), \tag{1}$$

where $\mathcal{C}_j^t \subseteq \mathcal{C}_j$ is the subset of patches in cluster $\mathcal{C}_j$ belonging to frame t and $\text{pos}(\mathbf{f}) \in \mathbb{N}^2$ is the (x, y) coordinate of patch $\mathbf{f}$ in its corresponding frame. Given the individual frame-wise cluster positions, we then compute the global positional encoding $\rho(\mathcal{C}_j)$ of cluster $\mathcal{C}_j$ as

$$\rho(\mathcal{C}_j) = \psi\left(\bigoplus_{t=1}^{T} \rho^t(\text{pos}(\mathcal{C}_j^t))\right), \tag{2}$$

where ρ^t is a frame-wise learnable positional encoding implemented as bilinear interpolation of parameter maps, $\oplus$ is the concatenation, and ψ is a learnable linear projection.

Each cluster centroid and its positional encoding act as a semantic prototype, which we use to initialize encoder queries. Each cluster $\mathcal{C}_j$ with centroid $\mathbf{c}_j \in \mathbb{R}^C$ constructs $i = 1, \ldots, N/K$ queries

$$\mathbf{q}_i^j = \mathbf{c}_j \oplus \rho(\mathcal{C}_j) \oplus \phi_i, \tag{3}$$

where ϕ_i is a learnable intra-cluster positional encoding to distinguish multiple queries per cluster.

Transformer Encoder. These queries $\mathcal{Q} = \bigoplus_{j=1}^{K} \{\mathbf{q}_i^j\}_{i=1}^{N/K}$, together with the V-JEPA 2.1 patch tokens $\mathcal{F}$, are the input to a Vision Transformer encoder with full self-attention (further details in Section A). By defining the queries as *input-dependent* cluster centroids of the patch tokens in contrast to commonly used *input-independent* positional encoding, we encourage query-patch attention based on semantic similarity instead of spatiotemporal position. Finally, we drop the output patch tokens and only use the transformed queries as our set of latents $\mathcal{Z} = \{\mathbf{z}_i\}_{i=1}^{N}$.

3.2 Decoding and Training

The decoder $\mathcal{D}$ reconstructs the original video given the fixed-size latent token set. In line with the ViT encoder, it is implemented as a ViT with full self-attention, taking the latent tokens $\mathcal{Z}$ and fixed 3D positionally encoded patch queries $\mathcal{P} = \{\mathbf{p}_i\}_{i=1}^{M}$ as input. As we directly decode to RGB patches, the decoder is trained end-to-end to decompress the object-centric 1D latent token set to a video with three spatiotemporal dimensions as well as to reconstruct RGB from the semantic V-JEPA 2.1 features. This puts semantic quality and consistency over exact RGB reconstruction, allowing us to achieve greater generative quality with few semantically meaningful tokens at the expense of slightly worse reconstruction of high-frequency details due to V-JEPA 2.1 bottleneck as feature extractor.

Following prior work [43], we train the autoencoder to minimize a composition of loss terms

$$\mathcal{L}_{\text{rec}} = \mathcal{L}_{\text{pix}} + \lambda_1 \mathcal{L}_{\text{LPIPS}} + \lambda_2 \mathcal{L}_{\text{KL}}\,, \tag{4}$$

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{rec}} + \lambda_3 \frac{\|\nabla \mathcal{L}_{\text{rec}}\|}{\|\nabla \mathcal{L}_{\text{adv}}\|} \mathcal{L}_{\text{adv}}\,, \tag{5}$$

where $\mathcal{L}_{\text{pix}}$ is an L1 pixel reconstruction loss, $\mathcal{L}_{\text{LPIPS}}$ is the LPIPS [44] loss, and $\mathcal{L}_{\text{KL}}$ is the KL divergence between the tokenizer encoder posterior (parameterized as a diagonal Gaussian) and the standard Gaussian prior. Furthermore, the adversarial loss $\mathcal{L}_{\text{adv}}$ is implemented using a discriminator consisting of a frozen DINOv2-S [30] backbone and a classification head trained along side. We set $\lambda_1 = 1.0$, $\lambda_2 = 10^{-5}$, $\lambda_3 = 0.2$, and use an additional adaptive weight for the adversarial loss based on its gradient norm in ratio to the combined other losses. Refer to Section A for more details.

3.3 Latent Generation in `TrackTok` Space

Unlike existing 1D tokenizers for videos [1, 12, 28], `TrackTok` learns a *set* of *continuous* latent tokens in contrast to a *sequence* of *discrete* ones. We train a latent rectified flow transformer [32] as the state-of-the-art generative paradigm for continuous spaces. Following conditional flow matching with the linear rectified flow noise schedule [45, 46], we construct noisy training examples

$$z_t = (1 - t) z_0 + t z_1, \tag{6}$$

where $t \in [0, 1]$ is the noise level sampled from a logit normal distribution [18], z_0 are the latent object-centric tokens obtained by encoding a sampled video from our training set, and $z_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is Gaussian noise. We train a latent video flow model v_θ, parameterized as a velocity field, to minimize the conditional flow matching loss

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}_{t,(z_0,c),z_1} \|v_\theta(z_t, t, c) - (z_1 - z_0)\|, \tag{7}$$

where c is a task-dependent conditioning signal. In class-conditional generation on UCF-101, c is the class label of the latent video z_0. In frame prediction on Kinetics-600, c is a set of *clean* latent tokens $z^c = \mathcal{E}(x^c)$ obtained by encoding the first 5 frames, with the same `TrackTok` encoder.

4 Experiments

In this section, we present our evaluation of `TrackTok`. Sec. 4.1 introduces our setup and Sec. 4.2 shows that our method presents a new frontier in efficient video encoding, well-suited for latent diffusion. In Sec. 4.3, we analyze our tokens w.r.t. semantic object alignment and temporal consistency and in Sec. 4.4 we ablate on our design choices. See App. A for full implementation details.

4.1 Experimental Setup

Datasets. We train our video tokenizers with token budget $N \in \{32, 64, 128, 256, 512, 1024\}$ on the combined Kinetics-600 [47] and UCF-101 [48] with $16 \times 128 \times 128$ resolution. Our evaluation consists of class-conditional generation on UCF-101 and frame prediction on Kinetics-600, following prior protocol [1, 10, 49]. We evaluate object centricity and semantic persistence of tokens on YouTube-VIS [17] with $N = 28$ tokens, matching the slot count of object-centric learning approaches [39].

Tokenizer. Our tokenizer uses a frozen V-JEPA 2.1 [16] ViT-L/16 with patch size $2 \times 16 \times 16$ and transformers as encoder and decoder with hidden size 768, 12 blocks, and 12 attention heads, trained from scratch. We perform k-Means clustering on V-JEPA2.1 features followed by a shared-identity multi-query pooling head with per-frame bilinearly-interpolated positional embeddings. Due to compute limitations, we train `TrackTok` for 100,000 steps (15 epochs). Note that LARP is trained for longer (75 epochs). For a fair comparison, we also report LARP with our training recipe.

Denoiser. For class-conditional video generation on UCF-101 and frame prediction on Kinetics-600, we train LightningDiT-XL [32] flow-matching denoisers on the frozen tokenizer's normalized latents for 100,000 steps at global batch size 512. At inference, we use a 50-step Euler sampler and a classifier-free guidance (CFG) scale of 3.0 for class-conditional generation only.

Token Impact Visualization. We visualize the spatiotemporal impact regions of individual tokens. Specifically, we replace one token at a time with an all-zeros token and measure how the pixel-wise video reconstruction error increases. After that, we obtain token impact masks by assigning each pixel to the token whose replacement resulted in the largest error increase for this particular pixel.

Unsupervised Object Discovery and Tracking. Building upon our token impact masks, we evaluate object alignment and temporal consistency on YouTube-VIS [17]. Object discovery is measured using three metrics: token purity, foreground ARI (FG-ARI) and mean best-overlap IoU (mBO), each frame scored independently and then averaged. We measure object tracking on the linked track map over the whole video with video-level FG-ARI, mBO, and $\mathcal{J}\&\mathcal{F}$. See App. B for metric descriptions.

4.2 A New Generation Quality and Compression Trade-Off

In both class-conditional generation on UCF-101 and frame prediction on Kinetics-600, our `TrackTok` provides a new frontier in efficient video generation (see Table 1, Fig. 3). On UCF-101, `TrackTok` maintains stable class-conditional generation quality across token budgets, achieving a comparable gFVD to that of LARP while using $8\times$ fewer tokens (128 vs. 1024). Even with just 32 tokens, `TrackTok` achieves a gFVD of 201, while LARP and AdapTok degrade sharply, reaching gFVDs of 672 and 1525, respectively. On Kinetics-600 dataset, our method outperforms LARP and AdapTok at equal token counts in the low-token regime, with $4.9\times$ and $10.8\times$ better gFVD, respectively. `TrackTok` at 128 tokens (40.7 gFVD) is competitive with OmniTok at 4096 tokens (32.9 gFVD), despite using $32\times$ fewer latent tokens.

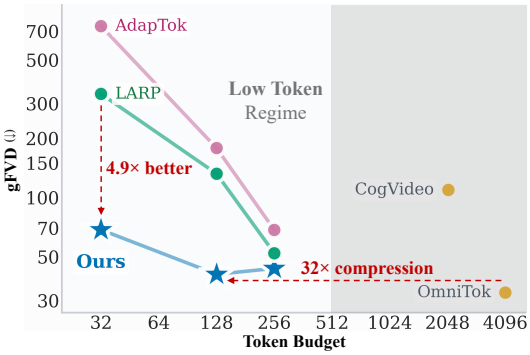


Figure 3: **Frame Prediction on Kinetics-600**, lower is better (gFVD). **Ours** outperforms baselines at low token regime.

We also evaluate reconstruction quality on UCF-101 of `TrackTok` and direct baselines LARP and AdapTok across all token budgets. As shown in Fig. 4 (left), our rFVD remains relatively stable as the token budget goes from 1024 down to 64 tokens. Meanwhile, both LARP and AdapTok degrade by an order of magnitude, and at 64 tokens `TrackTok` achieves $5.2\times$ lower rFVD than LARP. At 512 tokens and above, both baselines match or surpass `TrackTok`, which is expected since V-JEPA 2.1 features do not encode high-frequency RGB detail and thus bounds reconstruction fidelity.

Compression, i.e., token count reduction here, also translates directly into faster generation, since the generator's cost scales with sequence length. Fig. 4 (right) compares `TrackTok` against baselines at their native token budgets: `TrackTok` at 128 tokens generates a sample $8\times$ faster than LARP at 1024 tokens, at comparable quality. AdapTok and MAGVIT-v2 reach lower gFVD but take $18\times$ and $29\times$ longer per sample, respectively. The remaining baselines such as Cosmos-S, TATS, MAGVIT,

Table 1: **Class-Conditional Video Generation on UCF-101.** We report generation FVD (gFVD) across methods under comparable token budgets, token-to-pixel (T/P) ratios, training data size, and number of trainable parameters. While prior methods achieve lower gFVD at moderate compression rates, `TrackTok` remains competitive even under extreme compression (128 tokens, 0.012% T/P), highlighting a favorable efficiency–quality trade-off in the low-token regime. † denotes models trained using the same setup as ours. **Best** and second best are highlighted per token budget.

Method	#Tokens	T/P (%)	Data Size	#Params		gFVD↓
				Tokenizer	Generator	UCF-101
CV-VAE + DiT/XL [14]	4096	0.391	> 610M	182M	674M	262
OmniTok [15]	4096	0.391	1.4M	82.2M	650M	191
CogVideo [50]	2065	3.151	5.4M	–	9.4B	626
MAGVIT-v2-AR [49]	1280	0.488	1.8M	–	840M	109
TATS [51]	1024	0.391	< 0.5M	32M	321M	332
MAGVIT-AR [10]	1024	0.391	12M	158M	306M	265
AdapTok [2]	1024	0.391	< 0.5M	259M	633M	**67**
LARP [1]	1024	0.391	< 0.5M	173M	343M	107
LARP [1]†	1024	0.391	< 0.5M	173M	632M	102
`TrackTok`	1024	0.391	< 0.5M	173M	674M	117
Cosmos-S [52]	512	0.049	100M	112M	674M	191
AdapTok [2]	512	0.196	< 0.5M	259M	633M	**109**
LARP [1]†	512	0.196	< 0.5M	173M	632M	153
`TrackTok`	512	0.098	< 0.5M	173M	674M	116
AdapTok [2]	256	0.098	< 0.5M	259M	633M	241
LARP [1]†	256	0.098	< 0.5M	173M	632M	218
`TrackTok`	256	0.049	< 0.5M	173M	674M	**129**
AdapTok [2]	128	0.049	< 0.5M	259M	633M	460
LARP [1]†	128	0.049	< 0.5M	173M	632M	347
`TrackTok`	128	0.049	< 0.5M	173M	674M	**126**
AdapTok [2]	64	0.024	< 0.5M	259M	633M	823
LARP [1]†	64	0.024	< 0.5M	173M	632M	486
`TrackTok`	64	0.024	< 0.5M	173M	674M	**152**
AdapTok [2]	**32**	**0.012**	< 0.5M	259M	633M	1525
LARP [1]†	**32**	**0.012**	< 0.5M	173M	632M	672
`TrackTok`	**32**	**0.012**	< 0.5M	173M	674M	**201**

CV-VAE and OmniTokenizer are both slower (4–450×) and worse in generation quality (191–332 gFVD). `TrackTok` thus occupies the low-latency end of the Pareto front.

We further provide qualitative generation results in Fig. 5 and token visualizations in Fig. 6. With as few as 32 tokens, `TrackTok` maintains recognizable object structures and consistency over time.

Overall, `TrackTok` sets a new frontier for efficient tokenizers, achieving competitive generation quality with up to 8× fewer tokens and up to 8× faster generation than direct baselines on UCF-101.

4.3 Analysis of Object Alignment and Temporal Consistency

To analyze semantic object alignment and temporal consistency of our tokenization method, we evaluate `TrackTok`'s token impact masks in unsupervised object discovery and tracking on YouTube-VIS 2021, using metrics from object-centric representation learning (c.f. Sec. 4.1).

As shown in Table 2, token purity is high for both `TrackTok` and the baselines, with `TrackTok` achieving a further 10% improve-

	Obj. Discovery (per frame)			Tracking (across frames)		
	Purity↑	FG-ARI↑	mBO↑	FG-ARI↑	mBO↑	$\mathcal{J}\&\mathcal{F}$↑
AdapTok	83.1	6.2	10.5	2.7	8.0	7.6
LARP	84.5	6.9	12.3	3.3	8.5	7.7
Ours w/o CQ	84.9	9.9	13.7	6.3	10.7	10.5
Ours	**93.8**	**21.3**	**25.2**	**16.7**	**21.1**	**19.1**

Table 2: **Object Discovery and Tracking**. For equal token counts, TrackTok's tokens are more strongly aligned with individual objects as well as more temporally consistent than those of baselines.

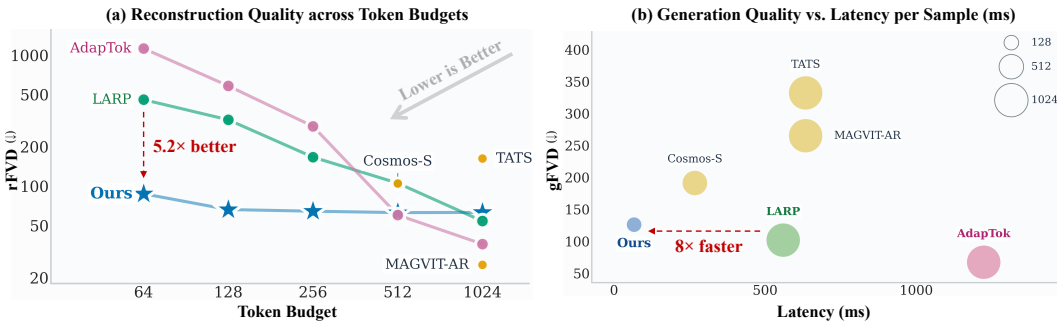


Figure 4: **Left: UCF-101 reconstruction quality**. Our rFVD remains stable while LARP and AdapTok significantly degrade at low token count. **Right: Generation quality vs. latency**, lower-is-better. Ours at 32 tokens achieves $8\times$ lower latency and comparable quality to LARP at 1024 tokens.

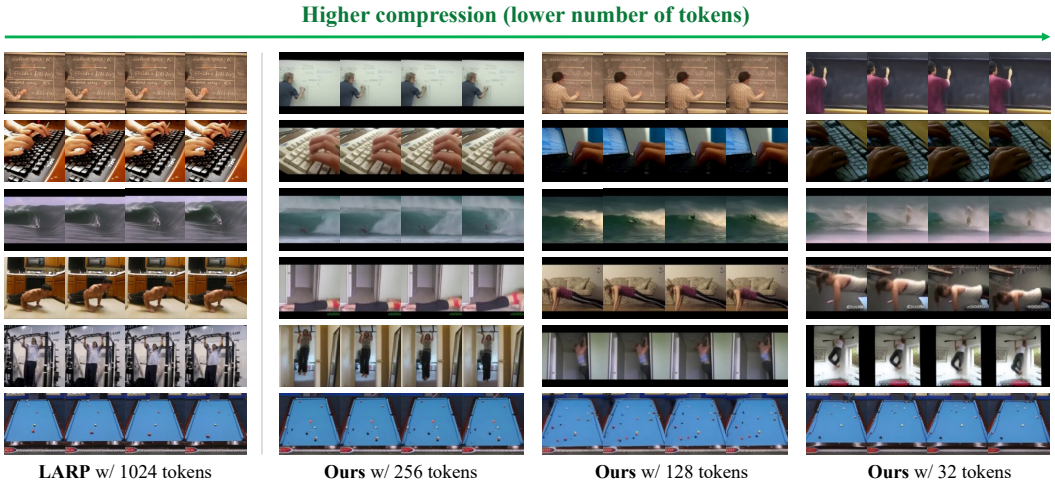


Figure 5: **Class-Conditional Generation on UCF-101 Across Token Budgets.** We show samples for diverse action categories, including hand motions (rows 1–2), dynamic natural scenes (row 3), and human body pose sequences (rows 4–5). Even at 32 tokens, our generated videos retain recognizable object structure, motion coherence, and semantic consistency across frames, comparable to LARP with $32\times$ as many tokens. We include more qualitative examples in Appendix E.

ment. This shows that most tokens roughly correspond to a single object. The baselines achieve this purity by fragmenting objects: their per-frame FG-ARI remains below 7%, and linking tokens across time reduces video FG-ARI to around 3%, as these fragments are re-assigned across clips. In contrast, `TrackTok` tokens consistently cover entire objects, achieving $6\times$ higher video FG-ARI and $2.5\times$ higher $\mathcal{J}\&\mathcal{F}$ compared to LARP and AdapTok. Intriguingly, removing our cluster queries (CQ) reduces token purity to the baseline level, halves object discovery performance, and decreases tracking scores by more than half. This highlights the importance of CQ for our design.

Fig. 7 compares our token impact visualization with ground-truth object masks. In the baselines, the tokens are spatially fragmented and do not correspond to semantic units. In contrast, `TrackTok` maintains a consistent cluster assignment for each runner throughout the sequence, preserving the same token identity. Refer to App. B for more qualitative examples.

In summary, we show that `TrackTok` indeed provides tokens with much richer semantics than existing state-of-the-art 1D tokenization methods. In combination with the previous results, this suggests that moving from input-independent, grid-based tokens to 1D set of input-dependent, semantic tokens enables much more efficient encoding for video generation.

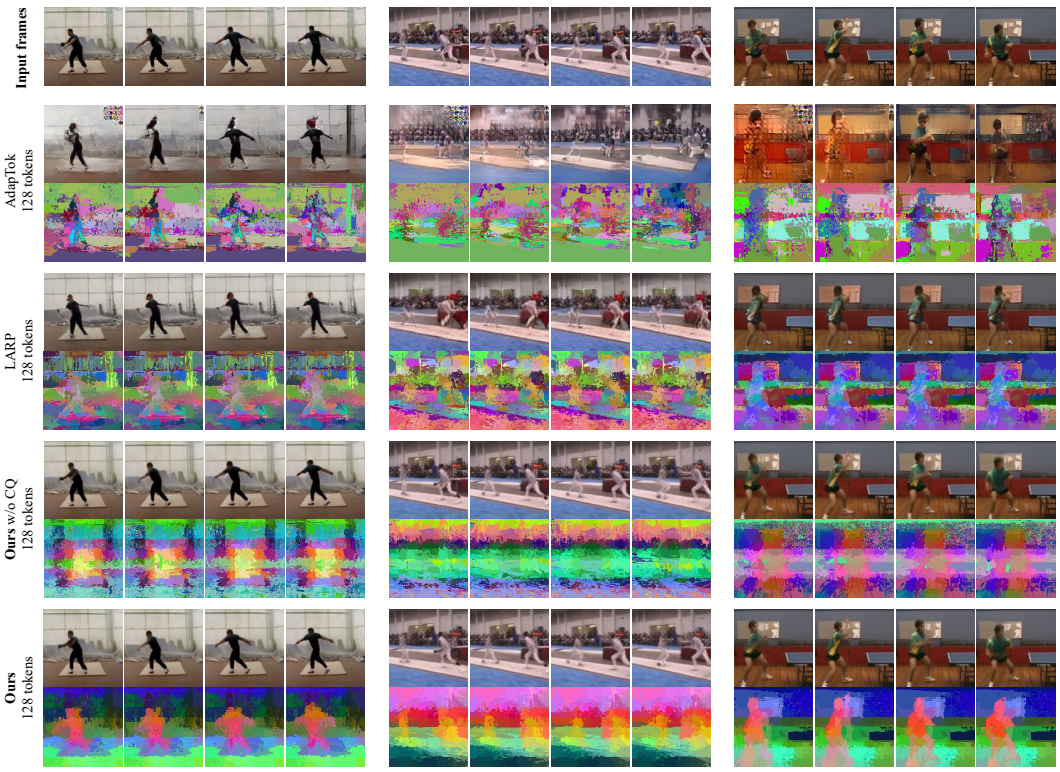


Figure 6: **Token Impact Across Time.** We sample four frames from the input clip together with reconstructed frames and token impact visualizations (c.f. Sec. 4.1). For each sample, we apply PCA on token features and map the top three components to RGB as color for the corresponding token impact mask. The visualization highlights the semantic object alignment and temporal tracking of tokens learned by `TrackTok` in contrast to AdapTok's or LARP's tokens, which are dominated by high-frequency textures and edges. Both baselines also show degraded reconstruction quality at 128 tokens. Furthermore, leveraging only V-JEPA 2.1 without our cluster queries (CQ) still results in tokens encoding fixed regions of the video. Refer to Appendix E for more uncurated examples.

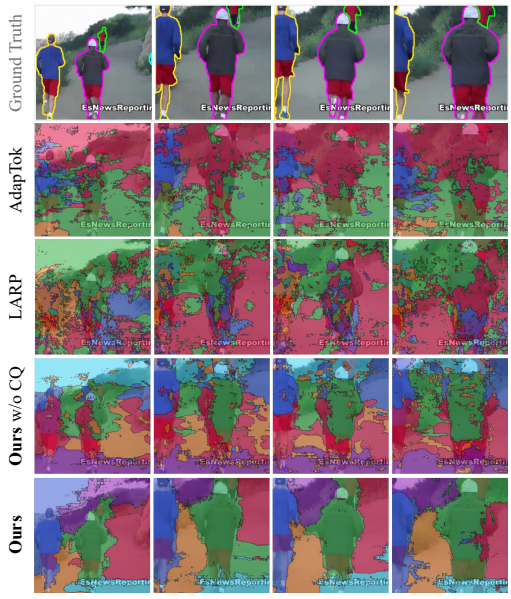
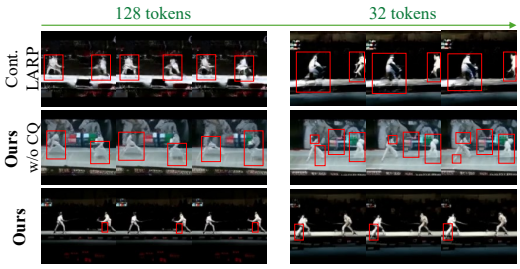


Figure 7: **Object Alignment & Temporal Consistency.** `TrackTok`'s tokens encode semantic units and track them consistently across frames.



(a) Qualitative ablation

Method	#Tokens	#Dim	LPIPS↓	rFVD↓	gFVD↓
Cont. LARP-L	32	128	0.0889	**78.61**	521.10
Ours w/o CQ	32	128	0.1400	95.47	236.06
Ours	32	128	0.1586	111.57	**200.58**
Cont. LARP-L	128	128	0.0487	**15.31**	555.28
Ours w/o CQ	128	128	0.1296	55.73	185.51
Ours	128	128	0.1411	63.89	**125.81**
Cont. LARP-L	256	64	0.0464	**13.68**	403.57
Ours w/o CQ	256	64	0.1264	50.93	129.14
Ours	256	64	0.1495	68.13	**128.41**

(b) Effect of CQ across token budgets

Figure 8: **Ablation Study.** We provide (a) qualitative samples of the `Fencing` class and (b) quantitative ablation of the encoder design.

4.4 Ablation Study

In Fig. 8, we provide qualitative and quantitative ablations of our two main architecture design choices. Continuous LARP-L is an adapted version without vector quantization and autoregressive prior and is therefore like `TrackTok` without V-JEPA 2.1 feature extractor and cluster queries. While `TrackTok` lacks behind in rFVD, mainly due to the bottleneck of V-JEPA 2.1 features not encoding high-frequency RGB details, we significantly improve generation quality. At high compression (32 tokens), Cont. LARP-L achieves lower rFVD but significantly degrades in generation quality, highlighting that reconstruction fidelity alone does not imply a well-structured latent space for generation. Similarly, removing our cluster queries (w/o CQ) degrades gFVD across all token counts, with the largest gap at low token budgets, showing that semantic object alignment that `TrackTok` learns is critical for generation quality under high compression. Refer to App. D for more ablations.

5 Conclusion

We proposed `TrackTok`, a novel video tokenizer that enables learning of *semantically persistent* tokens that capture and track semantic units over time. This represents a shift in how we encode videos, moving from spatially localized embeddings to holistic tokens using foundational features as inductive bias. We showed that `TrackTok` provides a new frontier for efficient encodings for video generation, with $8\times$ fewer tokens at similar generative quality compared to previous methods. It enables up to $8\times$ higher sampling speed at similar generative quality compared to state-of-the-art compression-focused tokenizer, thus enabling *an order of magnitude* faster video generation.

Limitations and Future Work. For future research on video tokenizers, we see `TrackTok`-based approaches as an interesting alternative to spatially localized holistic tokens. Our analysis suggests that such representations might provide a new angle on scaling video autoencoders to long videos. In addition to high compression and efficiency, we see potential in object tracking and video editing applications, which can profit from the emerging object-centric property. One limitation of introducing the object-centric property via soft inductive biases is the adaptive behavior in the cluster-query allocation: when scene content changes substantially within a clip, the encoder sometimes allows moving objects from one token to another, rather than forcing a single token to remain persistent. We view this as a deliberate degree of flexibility of the token representation (c.f. first row, Fig. E6 in supplement). A natural direction for future work is to make this allocation explicit, e.g., by tying token persistence to scene-level priors so that long-range identity is preserved more strictly when the downstream application requires it. In summary, we suggest a shift from traditional spatiotemporal video tokenizer towards 1D token sequences with semantic inductive biases that serves as a proof of principle for more scalable video generation.

Acknowledgements

We thank Xinya Chen for her valuable feedback and helpful discussions. Jan Eric Lenssen is supported by the German Research Foundation (DFG) - 556415750 (Emmy Noether Programme, project: Spatial Modeling and Reasoning).

References

[1] Hanyu Wang, Saksham Suri, Yixuan Ren, Hao Chen, and Abhinav Shrivastava. LARP: Tokenizing videos with a learned autoregressive generative prior. In *ICLR*, 2025.

[2] Yan Li, Changyao Tian, Renqiu Xia, Ning Liao, Weiwei Guo, Junchi Yan, Hongsheng Li, Jifeng Dai, Hao Li, and Xue Yang. Learning adaptive and temporally causal video tokenization in a 1d latent space. In *CVPR*, 2026.

[3] Zhen Xing, Qijun Feng, Haoran Chen, Qi Dai, Han Hu, Hang Xu, Zuxuan Wu, and Yu-Gang Jiang. A survey on video diffusion models. *ACM Comput. Surv.*, 57(2), November 2024.

[4] Tianwei Yin, Qiang Zhang, Richard Zhang, William T Freeman, Fredo Durand, Eli Shechtman, and Xun Huang. From slow bidirectional to fast autoregressive video diffusion models. In *CVPR*, 2025.

[5] Yoav HaCohen, Benny Brazowski, Nisan Chiprut, Yaki Bitterman, Andrew Kvochko, Avishai Berkowitz, Daniel Shalem, Daphna Lifschitz, Dudu Moshe, Eitan Porat, et al. Ltx-2: Efficient joint audio-visual foundation model. *arXiv preprint arXiv:2601.03233*, 2026.

[6] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

[7] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, Da Yin, Yuxuan.Zhang, Weihan Wang, Yean Cheng, Bin Xu, Xiaotao Gu, Yuxiao Dong, and Jie Tang. Cogvideox: Text-to-video diffusion models with an expert transformer. In *ICLR*, 2025.

[8] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

[9] Wenyi Hong, Ming Ding, Wendi Zheng, Xinghan Liu, and Jie Tang. Cogvideo: Large-scale pretraining for text-to-video generation via transformers. In *ICLR*, 2023.

[10] Lijun Yu, Yong Cheng, Kihyuk Sohn, José Lezama, Han Zhang, Huiwen Chang, Alexander G Hauptmann, Ming-Hsuan Yang, Yuan Hao, Irfan Essa, and Lu Jiang. MAGVIT: Masked generative video transformer. In *CVPR*, 2023.

[11] Xinlong Wang, Xiaosong Zhang, Zhengxiong Luo, Quan Sun, Yufeng Cui, Jinsheng Wang, Fan Zhang, Yueze Wang, Zhen Li, Qiying Yu, et al. Emu3: Next-token prediction is all you need. *arXiv preprint arXiv:2409.18869*, 2024.

[12] Zhentao Tan, Ben Xue, Jian Jia, Junhao Wang, Wencai Ye, Shaoyun Shi, Mingjie Sun, Wenjin Wu, Quan Chen, and Peng Jiang. Sweettok: Semantic-aware spatial-temporal tokenizer for compact video discretization. In *CVPR*, 2025.

[13] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

[14] 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. In *NeurIPS*, 2024.

[15] Junke Wang, Yi Jiang, Zehuan Yuan, Binyue Peng, Zuxuan Wu, and Yu-Gang Jiang. Omnitokenizer: A joint image-video tokenizer for visual generation. In *NeurIPS*, 2024.

[16] Lorenzo Mur-Labadia, Matthew Muckley, Amir Bar, Mahmoud Assran, Koustuv Sinha, Michael Rabbat, Yann LeCun, Nicolas Ballas, and Adrien Bardes. V-jepa 2.1: Unlocking dense features in video self-supervised learning. *arXiv preprint arXiv:2603.14482*, 2026.

[17] Linjie Yang, Yuchen Fan, Yang Fu, and Ning Xu. The 3rd large-scale video object segmentation challenge-video instance segmentation track. In *CVPR Workshop*, volume 5, 2021.

[18] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *ICLR*, 2024.

[19] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021.

[20] Anni Tang, Tianyu He, Junliang Guo, Xinle Cheng, Li Song, and Jiang Bian. Vidtok: A versatile and open-source video tokenizer. *arXiv preprint arXiv:2412.13061*, 2024.

[21] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *ICCV*, 2023.

[22] Yingqing He, Tianyu Yang, Yong Zhang, Ying Shan, and Qifeng Chen. Latent video diffusion models for high-fidelity long video generation. *arXiv preprint arXiv:2211.13221*, 2022.

[23] Michael Ryoo, AJ Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. Tokenlearner: Adaptive space-time tokenization for videos. *NeurIPS*, 34:12786–12797, 2021.

[24] Wilson Yan, Volodymyr Mnih, Aleksandra Faust, Matei Zaharia, Pieter Abbeel, and Hao Liu. Elastictok: Adaptive tokenization for image and video. In *ICLR*, 2025.

[25] Qihang Yu, Mark Weber, Xueqing Deng, Xiaohui Shen, Daniel Cremers, and Liang-Chieh Chen. An image is worth 32 tokens for reconstruction and generation. 2024.

[26] Roman Bachmann, Jesse Allardice, David Mizrahi, Enrico Fini, Oğuzhan Fatih Kar, Elmira Amirloo, Alaaeldin El-Nouby, Amir Zamir, and Afshin Dehghan. Flextok: Resampling images into 1d token sequences of flexible length. In *ICML*, 2025.

[27] Xin Wen, Bingchen Zhao, Ismail Elezi, Jiankang Deng, and Xiaojuan Qi. "principal components" enable a new language of images. In *ICCV*, 2025.

[28] Andrei Atanov, Jesse Allardice, Roman Bachmann, Oğuzhan Fatih Kar, Peter Fu, David Griffiths, Devon Hjelm, Afshin Dehghan, and Amir Zamir. VideoFlexTok: Flexible-length coarse-to-fine video tokenization. In *ICML*, 2026.

[29] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.

[30] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *TMLR*, 2024.

[31] Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. In *ICLR*, 2025.

[32] Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In *CVPR*, 2025.

[33] Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. In *ICLR*, 2026.

[34] Tianci Bi, Xiaoyi Zhang, Yan Lu, and Nanning Zheng. Vision foundation models can be good tokenizers for latent diffusion models. In *CVPR*, 2026.

[35] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In *NeurIPS*, 2020.

[36] Thomas Kipf, Gamaleldin Fathy Elsayed, Aravindh Mahendran, Austin Stone, Sara Sabour, Georg Heigold, Rico Jonschkowski, Alexey Dosovitskiy, and Klaus Greff. Conditional object-centric learning from video. In *ICLR*, 2022.

[37] Gamaleldin Elsayed, Aravindh Mahendran, Sjoerd van Steenkiste, Klaus Greff, Michael C Mozer, and Thomas Kipf. Savi++: Towards end-to-end object-centric learning from real-world videos. In *NeurIPS*, 2022.

[38] Gautam Singh, Yi-Fu Wu, and Sungjin Ahn. Simple unsupervised object-centric learning for complex and naturalistic videos. In *NeurIPS*, 2022.

[39] Anna Manasyan, Maximilian Seitzer, Filip Radovic, Georg Martius, and Andrii Zadaianchuk. Temporally consistent object-centric learning by contrasting slots. In *CVPR*, pages 5401–5411. IEEE, 2025.

[40] Maximilian Seitzer, Max Horn, Andrii Zadaianchuk, Dominik Zietlow, Tianjun Xiao, Carl-Johann Simon-Gabriel, Tong He, Zheng Zhang, Bernhard Schölkopf, Thomas Brox, and Francesco Locatello. Bridging the gap to real-world object-centric learning. In *ICLR*, 2023.

[41] Andrii Zadaianchuk, Maximilian Seitzer, and Georg Martius. Object-centric learning for real-world videos by predicting temporal feature similarities. In *NeurIPS*, 2023.

[42] Görkay Aydemir, Weidi Xie, and Fatma Guney. Self-supervised object-centric learning for videos. In *NeurIPS*, 2023.

[43] Weiliang Chen, Yuanhui Huang, Xuebo Wang, and Yueqi Duan. Tivtok: Broadcasting time-invariant tokens for scalable video tokenization. *arXiv preprint arXiv:2606.17590*, 2026.

[44] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[45] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *ICLR*, 2023.

[46] Xingchao Liu, Chengyue Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *ICLR*, 2023.

[47] Joao Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. A short note about kinetics-600. *arXiv preprint arXiv:1808.01340*, 2018.

[48] 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.

[49] Lijun Yu, Jose Lezama, Nitesh Bharadwaj Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Agrim Gupta, Xiuye Gu, Alexander G Hauptmann, Boqing Gong, Ming-Hsuan Yang, Irfan Essa, David A Ross, and Lu Jiang. Language model beats diffusion - tokenizer is key to visual generation. In *The Twelfth International Conference on Learning Representations*, 2024.

[50] Wenyi Hong, Ming Ding, Wendi Zheng, Xinghan Liu, and Jie Tang. Cogvideo: Large-scale pretraining for text-to-video generation via transformers. *arXiv preprint arXiv:2205.15868*, 2022.

[51] Songwei Ge, Thomas Hayes, Harry Yang, Xi Yin, Guan Pang, David Jacobs, Jia-Bin Huang, and Devi Parikh. Long video generation with time-agnostic vqgan and time-sensitive transformer. In *ECCV*, 2022.

[52] Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025.

[53] Mahmoud Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

[54] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *NeurIPS*, pages 10078–10093, 2022.

[55] Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

[56] Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, et al. Qwen-image technical report. *arXiv preprint arXiv:2508.02324*, 2025.

TrackTok: Object-Centric Video Tokenization with Semantically Persistent Tokens

Appendix

This supplement provides additional technical details, experimental setup, ablations, and qualitative results supporting our work on efficient video encoding for generative modeling. We **strongly encourage** readers to review our token impact analysis (Sec. C) which highlights the difference in our token representation compared to baselines. We also provide **uncurated** additional qualitative examples for both reconstruction and generation in Sec. E.

(A) Additional Implementation Details **16**

(A.1) Tokenizer and Denoiser Training Details 16

(A.2) Token-to-Pixel Ratio 18

(A.3) Latency Measurement 18

(B) Unsupervised Object Discovery and Tracking **20**

(B.1) Evaluation Protocol and Metrics 20

(B.2) Hyperparameters 21

(C) Token Impact Analysis **22**

(C.1) Visualization Details 22

(C.2) Token Heatmap Comparison 22

(D) Additional Ablations & Comparisons **28**

(D.1) Positional Encoding of Cluster Queries 28

(D.2) Representation Autoencoder as Encoder 28

(D.3) Training-Free Token Reduction of V-JEPA 2.1 Features 28

(D.4) Number of Clusters K 29

(D.5) Token Dimension 29

(D.6) Comparision with Text-Based Semantic Baseline 30

(E) Qualitative Examples **31**

A Implementation Details

A.1 Tokenizer and Denoiser Training Details

We provide a detailed description of the training recipes used for the `TrackTok` tokenizer and the LightningDiT-XL denoiser. All experiments are run on $16\times128\times128$ video clips. Hyperparameters that we share across all token budgets are listed once, and those that differ across the token configurations are reported per configuration. We train our tokenizers on 2–4 NVIDIA H100 GPUs with global batch size 64 ($\sim$36–61 hours), and our denoisers on 4–8 H100 GPUs, dependent on the token count, with global batch size 512 ($\sim$40–66 hours). Please refer to Tab. A1 and Tab. A2 for precise summary.

Datasets. For a direct and fair comparison with baselines LARP [1] and AdapTok [2], we train both our tokenizers and denoisers with the exaxt same data as these baselines. In specific, all our tokenizers on the combined dataset of UCF-101 and Kinetics-600 at $16\times128\times128$ with random horizontal flipping. For class-conditional generation, we train our denoisers only on UCF-101 dataset; for frame predicition, we train our denoisers only on Kinetics-600 dataset. For evaluation, we use the validation split of each dataset at the same resolution, with center cropping.

A.1.1 Autoencoder Training

We first train the video tokenizers to obtain the latent tokens for denoiser training.

Architecture. Our encoder and decoder designs consist each of 12-layer transformers with hidden size 768 and 12 heads. Given the input video of 16 frames, the semantic features are extracted from a frozen V-JEPA2.1 ViT-L/16 at the 128×128 resolution with patch size $2\times16\times16$. We perform k-Means clustering with 20 Lloyd iterations on the V-JEPA2.1 features and pool each cluster with a shared-identity multi-query head: each query concatenates the cluster centroid, per-frame bilinearly interpolated positional embeddings of dimension d_{PE}=64, and a learned identity embedding of dimension d_{id}=32 that is shared across clusters. We vary the token budget by jointly adjusting the number of clusters K, the number of query tokens per cluster, and the bottleneck dimension d. For budgets of 32, 64, and 128 tokens, we fix K=32 and use 1, 2, and 4 query tokens per cluster, respectively, with d=128. For larger budgets of 256, 512, and 1024 tokens, we progressively reduce K to 16, 8, and 4, while increasing the number of query tokens per cluster to 16, 64, and 256, and reducing d to 64, 32, and 16, respectively. This allows us to explore different allocations of representational capacity across number of clusters, query tokens, and feature dimensions under varying token budgets. Refer to Section D for a quantitative ablation.

Optimization. We use AdamW with β=(0.9, 0.95), weight decay 10^{-4}, peak learning rate 10^{-4}, and a OneCycleLR schedule (cosine annealing, 5% warm-up over 100,000 steps). Gradients are clipped to norm 1. Training runs for 100,000 steps. The discriminator is optimized identically (AdamW, β=(0.9, 0.95), weight decay 10^{-4}, learning rate 10^{-4}, gradient norm clip 1).

Losses. The reconstruction objective combines ℓ_1 pixel loss (weight 1.0), LPIPS-VGG (weight 1.0), and a KL regularizer (weight 10^{-5}). The adversarial loss is a non-saturating GAN against a frozen DINOv2-S/14 discriminator, which is enabled at step 30,000 with adversarial weight 0.2 on the generator (adaptive weight balancing turned on), LeCAM regularization weight 0.001, and a discriminator update frequency of 1:5 relative to the generator. This follows the loss design in [43].

A.1.2 Denoiser Training

We further describe our denoiser setup for class-conditional generation and frame prediction tasks.

Architecture. We train a LightningDiT-XL [32] flow-matching denoiser with hidden size 1152, 28 transformer blocks, and 16 attention heads. Following Table 7 of [32] we use only absolute (frequency) positional embeddings; the rotary relative positional embedding is disabled. The denoiser operates directly on the frozen tokenizer's normalized latents, where the normalization statistics are precomputed per tokenizer on the UCF-101 training split for class-conditional generation, and on Kinetics-600 training subset for frame prediction.

Table A1: **Autoencoder Training Configuration.** Values are shared across all six token budgets unless listed per budget.

Component	Setting
Architecture	
Backbone (frozen)	V-JEPA2.1 ViT-L/16, patch $2\times16\times16$, input 128×128
Encoder / decoder	Transformer, hidden size 768, 12 layers, 12 heads
Decoder output patch	$4\times8\times8$
Pooling head	shared-identity multi-query; query $=$ [centroid $\|\|$ PE $\|\|$ identity]
K-means iterations	20
Positional emb.	framewise bilinear, d_{PE}=64
Identity emb.	d_{id}=32, shared across clusters
Bottleneck	Gaussian (KL-regularized)
# Tokens N	32 / 64 / 128 / 256 / 512 / 1024
Clusters K	32 / 32 / 32 / 16 / 8 / 4
Queries per cluster	1 / 2 / 4 / 16 / 64 / 256
Bottleneck dim. d	128 / 128 / 128 / 64 / 32 / 16
Trainable params (G / D)	173M / 0.15M
Discriminator	DINOv2-S/14, frozen backbone
Optimization	
Optimizer (G / D)	AdamW, weight decay 10^{-4}
Generator LR	warm-up from 4×10^{-6} to 10^{-4} over the first 5k steps, then cosine decay
Generator Adam β	β_1 varied inversely with the LR ($0.95 \to 0.85 \to 0.95$), β_2=0.95
Discriminator LR	constant 10^{-4}
Discriminator Adam β	$(0.9,\ 0.95)$
Gradient clipping (G / D)	norm 1.0
Generator steps	100,000
Global batch size	64
Precision	FP32 (TF32 matmuls)
Losses	
ℓ_1 pixel	weight 1.0
LPIPS (VGG)	weight 1.0
KL	weight 10^{-5}
GAN	non-saturating, noisy label smoothing
Adversarial weight	$0.2\times$ adaptive (VQGAN-style) weight
GAN start step	30,000
D update frequency	once per 4 generator steps
LeCAM weight	10^{-3}
R_1 gradient penalty	0
Spectral norm.	off
Data	
Datasets (train)	combined UCF-101 + Kinetics-600
Clips	16 consecutive frames, native frame rate
Preprocessing	resize + center crop to 128×128
Augmentation	random horizontal flip

Flow-matching objective. We use the rectified-flow parameterization (u_t prediction, MSE loss) with logit-normal time sampling (μ=2.0). Class conditioning is dropped with probability 0.1 during training to enable classifier-free guidance.

Optimization. AdamW with β=$(0.9,\ 0.95)$, weight decay 0, learning rate 10^{-4}, 100,000 steps, global batch size 512 (4 GPUs $\times$ batch size 128), and no exponential moving average. We follow the LightningDiT recipe and do not apply learning-rate scaling.

Sampling. For both tasks, we sample with a deterministic 50-step Euler solver on the rectified-flow ODE, using equal step sizes from t=1 (noise) to t=0 (data) and no noise injection between steps. We use classifier-free guidance with a constant scale of 3.0. For class-conditional generation on UCF-101, guidance on the class label is applied at every step. For frame prediction on Kinetics-600, the denoiser is conditioned on the first 5 frames, and guidance on this frame condition is applied

only for $t \leq 0.9$, i.e., not during the first, noisiest 10% of the trajectory. All FVDs use the 400-class I3D features of prior work. Class-conditional FVD is computed between $10{,}000$ generated clips and $10{,}000$ real clips from the UCF-101 training split at 128×128; frame-prediction FVD is computed on $50{,}000$ clips from the Kinetics-600 validation split, with all frames downsampled to 64×64.

A.2 Token-to-Pixel Ratio

Note on resolution mismatch. A baseline we compare against may report in their original papers at video resolutions that differ from our $16\times128\times128$ setting. Comparing token counts directly across these settings would be misleading, since a tokenizer that emits, e.g., 1024 tokens at 256×256 is operating at a substantially *higher* compression ratio than one emitting the same 1024 tokens at 128×128. To make compression regimes comparable across publications, we additionally report each method's *token-to-pixel ratio*

$$\mathrm{T/P} \;=\; \frac{N_{\text{tokens}}}{T \cdot H \cdot W},$$

where N_{tokens} is the number of latent tokens produced for a clip of T frames and spatial size $H\times W$. Smaller $\mathrm{T/P}$ corresponds to stronger compression. This ratio captures the intuitive notion of "tokens per pixel of input video" and lets us place each baseline on the same compression axis as our 32/128/256-token `TrackTok` configurations, which at $16\times128\times128$ correspond to $\mathrm{T/P} \approx 1.22\times10^{-4}$, 4.88×10^{-4}, and 9.77×10^{-4}, respectively. We retain each baseline's reported reconstruction and generation quality at its native evaluation resolution and only normalize the token budget through $\mathrm{T/P}$; we do not re-train baselines at 128×128.

A.3 Latency Measurement

Hardware and software. All latency numbers reported in Tab. 1 are measured on a single NVIDIA H100 NVL (94 GB). Baseline methods and `TrackTok` run in the same numerical regime: fp32 master weights under a bfloat16 `autocast` (PyTorch Lightning's `bf16-mixed`), with TF32 enabled for the remaining fp32 matmuls and, for the autoregressive (AR) models, the KV cache held in bfloat16. Here, we isolate the cost of the generator only.

Protocol. We run one warm-up batch followed by $M{=}3$ measured batches at batch size B. Each measured batch is bracketed by `torch.cuda.synchronize()` before reading `perf_counter()` at its start and at its end. The reported per-sample latency is the mean batch wall-clock time divided by the batch size,

$$\text{Latency} \;=\; \frac{1}{M}\sum_{b=1}^{M}\frac{t_b}{B},$$

where t_b is the wall-clock time of the b-th measured batch. We use the same $B{=}64$ for every method, including all of our denoisers, with two exceptions forced by memory: OmniTokenizer (4096 tokens) and CV-VAE (4096 tokens at 256^2) run at $B{=}16$.

Reproducibility. We use a fixed seed of 0, but the measurement is insensitive to the random seed. On an H100, the three measured batches for each denoiser vary within 1.2% from their mean.

Table A2: **Denoiser (LightningDiT-XL) Training Configuration.** Settings spanning both columns are shared by class-conditional generation (UCF-101) and frame prediction (Kinetics-600). The recipe is the same for every token budget; only the sequence length N and latent dimension d (Tab. A1) differ, plus the batch size and number of steps for $N{=}1024$.

Component	**Class-conditional**	**Frame prediction**
Architecture		
Backbone	LightningDiT-XL: hidden size 1152, 28 layers, 16 heads	
Block	SwiGLU MLP (ratio $8/3$), RMSNorm, adaLN-Zero	
Token embedding	linear projection of each latent token ($d \to 1152$)	
Positional embedding	fixed sinusoidal, absolute (no RoPE)	
Timestep embedding	sinusoidal, 256-dim	
Condition	class label (101 classes), added to the timestep embedding	latents of the first 5 frames, prepended as N clean tokens
Sequence length	N	$2N$ (condition + target)
Dropped condition	learned null class embedding	learned null tokens
Optimization		
Optimizer	AdamW, $\beta{=}(0.9,\ 0.95)$, weight decay 0	
Learning rate	constant 10^{-4}, no warm-up	
Gradient clipping	norm 1.0	
Global batch / steps	512 / 100k; 256 / 200k for $N{=}1024$	
Precision	bf16 mixed	
EMA	none	
Flow matching		
Objective	rectified flow, velocity prediction, MSE loss	
Timestep sampling	logit-normal, $\mu{=}2.0$, $\sigma{=}1.0$ (biased toward high noise)	
Condition dropout	10% of training samples	
Latent normalization	per-channel mean/std of the training-set latents	
Sampling		
Solver	deterministic Euler, 50 uniform steps	
Classifier-free guidance	scale 3.0 at every step	none
Data and evaluation		
Training data	UCF-101 (train)	Kinetics-600 (train)
Clips	16 consecutive frames, resize + center crop to $128{\times}128$, no flip	
FVD evaluation	10k clips vs. UCF-101 train, $128{\times}128$	50k clips, K600 val, $64{\times}64$
FVD features	I3D (400 classes)	

B Unsupervised Object Discovery and Tracking

We test whether tokens bind to objects and keep their identity over time. All tokenizers are frozen and probed without any supervision or fine-tuning: we derive a segmentation from each token's influence on the reconstruction, link tokens across clips with a training-free procedure, and score the result against ground-truth instance masks. Refer to Section E for additional qualitative visualizations.

B.1 Evaluation Protocol and Metrics

Data. We use YouTube-VIS 2021 [17]. Since the official validation split has no public masks, we follow SlotContrast [39] and hold out the last 210 training videos for evaluation (12–62 frames, median 36; 1–25 annotated objects, median 2). The remaining 2,775 videos are used to train the tokenizers and to tune the linking hyperparameters.

Tokenizers. `TrackTok`, LARP [1], and AdapTok [2] are trained on the same 2,775 videos with the same budget: 28 tokens of dimension 64 per 4-frame clip at 518×518, for $\sim$100k steps. `TrackTok` uses a frozen V-JEPA2.1 ViT-B/16 backbone with $K=28$ clusters and one query per cluster. LARP and AdapTok use vector quantization with a 8,192-entry codebook. With 4-frame clips, AdapTok's block-causal allocation reduces to a single block, and we evaluate it at its full budget of 28 tokens.

From tokens to masks. Each video is split into consecutive, non-overlapping 4-frame clips (the last clip is padded by repeating its final frame, and padded frames are not scored), and each clip is tokenized independently. To assign pixels to tokens, we ablate each token in turn by setting it to zero, decode, and assign every pixel to the token whose removal changes its reconstruction the most (ℓ_2 error). This measures what each token encodes after decoding rather than how the encoder groups its inputs, and is applied identically to all three tokenizers.

Linking tokens across clips. Token identities are only meaningful within a clip, so we link tokens causally across clips. For each token, we consider the tokens of the preceding clip whose mask centroids lie within a radius r, take the k most similar by cosine similarity of their token embeddings (for `TrackTok`, the 64-d bottleneck latents; for LARP and AdapTok, the dequantized embeddings passed to the decoder), and let them vote for their tracks with weights $\mathrm{softmax}(\mathrm{sim}/\tau)$. The token joins the winning track if its peak similarity exceeds a threshold, and starts a new track otherwise. Because 28 tokens usually exceed the number of objects, an optional second stage merges whole tracks by average-linkage agglomerative clustering of their mean embeddings (cosine distance), cut at a distance threshold and/or a maximum number of clusters. These hyperparameters are tuned separately for each tokenizer by a grid search on the training videos at 128×128, maximizing whole-video FG-ARI. Final results are computed on the 210 held-out videos at 518×518.

Metrics. We derive segmentations from the tokens by assigning each pixel to its owning token via token ablation. Each frame is thus partitioned into at most 28 token regions; pixels with overlapping ground-truth instances are ignored. We report FG-ARI and mBO following SlotContrast [39], both per frame and over the entire video, as well as $\mathcal{J}\&\mathcal{F}$ and token purity. Per-frame scores are averaged over frames, then all metrics are averaged over videos.

- **Token purity** measures how consistently a token represents a single object. For a token, we compute the fraction of its foreground pixels belonging to its dominant object, then average across tokens weighted by their foreground area. For example, a token covering 90% object 1 and 10% object 2 has 90% purity. Purity is computed on raw per-clip tokens and does not penalize fragmentation: splitting an object across many small tokens can still yield high purity.
- **Object discovery** measures whether tokens capture whole objects within individual frames, without linking identities across clips. Token IDs are local to each 4-frame clip, and frames without annotated objects are excluded. *FG-ARI* measures clustering agreement between token IDs and ground-truth instance IDs on foreground pixels (0 indicates chance-level agreement; 1 indicates a perfect match). It penalizes both splitting an object across tokens and merging multiple objects into one token, regardless of how tokens are assigned to the background. *mBO* computes the IoU between each ground-truth object and its best-

matching token mask, including background pixels, then averages over objects. Matching is not one-to-one, so mBO measures how completely a single token covers each object.

- **Object tracking** measures whether token identities remain consistent across frames. After linking and merging, tokens receive global track IDs, producing a single label map for the entire video. *Video FG-ARI* evaluates clustering agreement across all foreground pixels in all frames, penalizing object fragmentation, merging, and identity switches. *Video mBO* matches each object to its best-overlapping track across the video and averages IoU over objects. $\mathcal{J}\&\mathcal{F}$ evaluates region overlap ($\mathcal{J}$, IoU) and boundary accuracy ($\mathcal{F}$). We average both measures over annotated frames for each object–track pair, then use Hungarian matching to assign tracks to objects based on their mean $(\mathcal{J}+\mathcal{F})/2$; unmatched objects score zero. Matching is performed once per video, requiring each track to follow its object throughout the video for a high score.

B.2 Hyperparameters

Table B1: **Token-linking hyperparameters used for the object discovery and tracking evaluation.** Tokens are linked across consecutive 4-frame clips by cosine similarity of their bottleneck embeddings, then tracks are merged agglomeratively into object-level clusters. For every method the full joint grid (540 linking $\times$ 54 merge combinations) is swept on the YTVIS-2021 training split at 128 px, ranked by whole-video FG-ARI, and the best configuration is evaluated once on the 210 validation videos at 518 px. "Ours w/o CQ" reuses the configuration selected for the full model. "–" denotes a disabled component.

Hyperparameter	**Search grid**	**AdapTok**	**LARP**	**Ours w/o CQ**	**Ours**
Token linking (across clips)					
Context length L (clips)	$\{1, 2, 4, 8\}$	1	1	1	1
Neighbour radius r	$\{0.2, 0.35, -\}$	0.2	0.2	0.35	0.35
Top-k votes	$\{1, 3, 5\}$	3	5	5	5
Temperature τ	$\{0.05, 0.1, 0.2\}$	0.05	0.2	0.1	0.1
Link threshold θ_{link}	$\{0.3, 0.45, 0.6, 0.75, 0.9\}$	0.3	0.3	0.9	0.9
Track merging (object-level clustering)					
Merge threshold θ_{merge}	$\{-, 0.1, 0.15, \ldots, 0.4, 0.5\}$	0.5	0.2	–	–
Max. clusters K_{merge}	$\{-, 2, 3, \ldots, 10\}$	8	–	6	6

Table B2: **Evaluation settings shared by all tokenizers** in the object discovery and tracking benchmark.

Setting	**Value**
Dataset / split (hyperparameter selection)	YTVIS-2021 train, 128 px, without $\mathcal{J}\&\mathcal{F}$
Dataset / split (evaluation)	YTVIS-2021 val (210 videos), 518 px
Clip length	4 frames
Tokens per clip / token dim.	28 / 64
Token→pixel assignment	Single-token ablation, per-pixel arg-max of reconstruction error
Linking feature	Bottleneck token embedding (cosine similarity)
Merge linkage	Average linkage on mean track embedding (cosine distance)
Selection criterion	Whole-video FG-ARI on the training split

C Token Impact

C.1 Visualization Details

In section 4.1, we described how to visualize the token impact on high level. To probe what each latent token encodes, we conduct two analyses:

(i) *Per-token impact per sample*: for a fixed input video we encode once, then re-decode after zeroing a single token at a time, yielding a per-pixel L_2 error map for every token. We assign each pixel to the token whose removal most affects it and color the resulting impact map by projecting each token's latent vector onto the top-three principal components of the latent set. This makes both the spatial support of a token and the similarity structure between tokens visible in a single frame.

(ii) *Per-token impact across samples*: for each token $\mathbf{z}_i$, we average its per-pixel ablation error across all clips in the validation split, producing average token-indexed heatmaps.

C.2 Token Heatmap Comparison

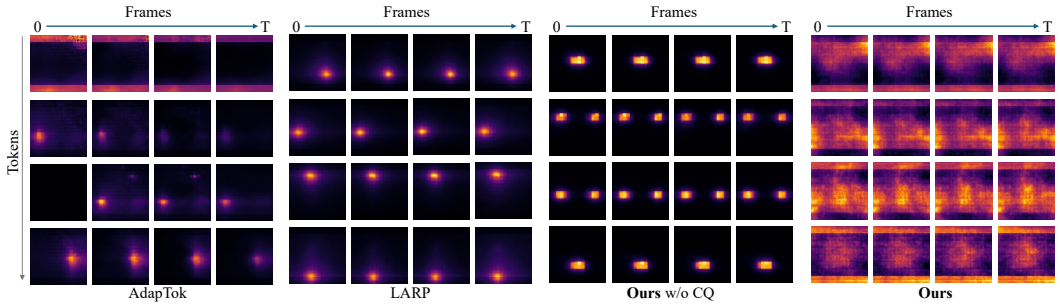


Figure C1: **Average Token Heatmap Videos.** We visualize the impact of individual tokens on video pixels, averaged over the UCF-101 validation split. Each row shows a randomly selected token, independently per method. AdapTok and LARP (left) learns mostly tokens that encode individual video patches. Encoding V-JEPA 2.1 features instead of RGB, ours without cluster queries (middle) learns to encode the same spatial patches for all frames into shared tokens. This encoder behavior changes drastically by adding cluster queries as additional inductive bias, resulting in `TrackTok` (right) to assign tokens to semantic units depending on the input. Full visualizations in below.

We examine whether tokens consistently bind to semantic regions over time using the two complementary ablation-based visualizations introduced in App. C.1. A token that consistently encodes the same spatial region produces a sharp, localized blob, whereas a token whose role varies across inputs produces a more diffuse map. This provides a direct measure of each tokenizer's spatial bias.

We extend the per-token impact analysis in Sec. 4.3 to the *dataset level* in Fig. C1. For each tokenizer, we encode every clip in the UCF-101 validation split and re-decode it after zeroing each token $\mathbf{z}_i$, yielding a per-pixel L_2 ablation error. Averaging these errors across the validation set produces one influence map per token, capturing *where it tends to act independently of the input scene*. We visualize the tokens as a 32×16 grid (rows: tokens; columns: frames). For both LARP and AdapTok with released checkpoints of 1024 tokens, we sample 32 of its tokens. For our two 32-token configurations, with and without the cluster-query head, we visualize the complete token set.

LARP (Fig. C2). LARP's aggregate maps decompose the 128×128 canvas into an almost regular tile pattern: each token's influence is concentrated in a sharp, near-stationary blob, and even within this small subset of LARP's 1024 tokens the blobs already tile a fraction of the frame, and the full set tiles it densely. Token identity in LARP is therefore effectively a coordinate, not a semantic descriptor, which is consistent with LARP using a fixed set of learned positional queries to read out a transformer feature grid.

AdapTok (Fig. C3). Similar to LARP, AdapTok tokens are spatially localized: almost every token's heatmap collapses to one or two point-like blobs that remain fixed across all reachable frames. The

main difference is temporal and arises from the architecture rather than being learned. AdapTok divides tokens into four causal blocks, each corresponding to four frames, with block b tokens affecting only frames $4b$ onward. This produces a staircase pattern in the token-frame grid, with each block blank before its corresponding frame group.

Ours, ablating the cluster-query (CQ) head (Fig. C4). Replacing k-Means cluster-query bottleneck with a plain learned-query encoder of the same depth, width, and token budget recovers the same tile-like structure as LARP. This is informative on its own: the location-encoding behavior is not a property of the transformer encoder-decoder, of the GAN training, or of the latent dimensionality; it is the default outcome whenever the tokenizer is free to assign each query a fixed positional role. It also lets us isolate the effect of our cluster-query bottleneck without conflating it with backbone or training-recipe changes.

Ours (Fig. C5). With the cluster-query bottleneck enabled, each token's average influence is spread broadly across the frame rather than concentrated on a fixed region. Because the average is taken over all validation videos, this can only happen if the spatial support of a given token *varies from clip to clip*, which means that tokens are allocated based on scene content rather than on their index. This is the dataset-level counterpart of the per-sample visualization in Fig. 6, where individual tokens cover semantically coherent objects whose spatial extent depends on the scene.

The three figures together demonstrates that the cluster-query head is what turns the latent tokens from a fixed spatial grid into a content-allocated set, and that this is the mechanism behind the favorable trade-off our tokenizer achieves at extreme compression. We use the aggregate maps only as a qualitative diagnostic for this hypothesis; the quantitative case for the design is made by the gFVD at lower token budgets and the throughput numbers in Tab. 1.

Figure C2: **Average token heatmaps for LARP [1].** Most heatmaps are sharp blobs that stay anchored to the same image region across frames and across videos, recovering an obvious tile pattern that tiles the 128×128 frame. This confirms that most of LARP's tokens are *positionally* bound: token identity is tied to a fixed spatial location rather than to scene content.

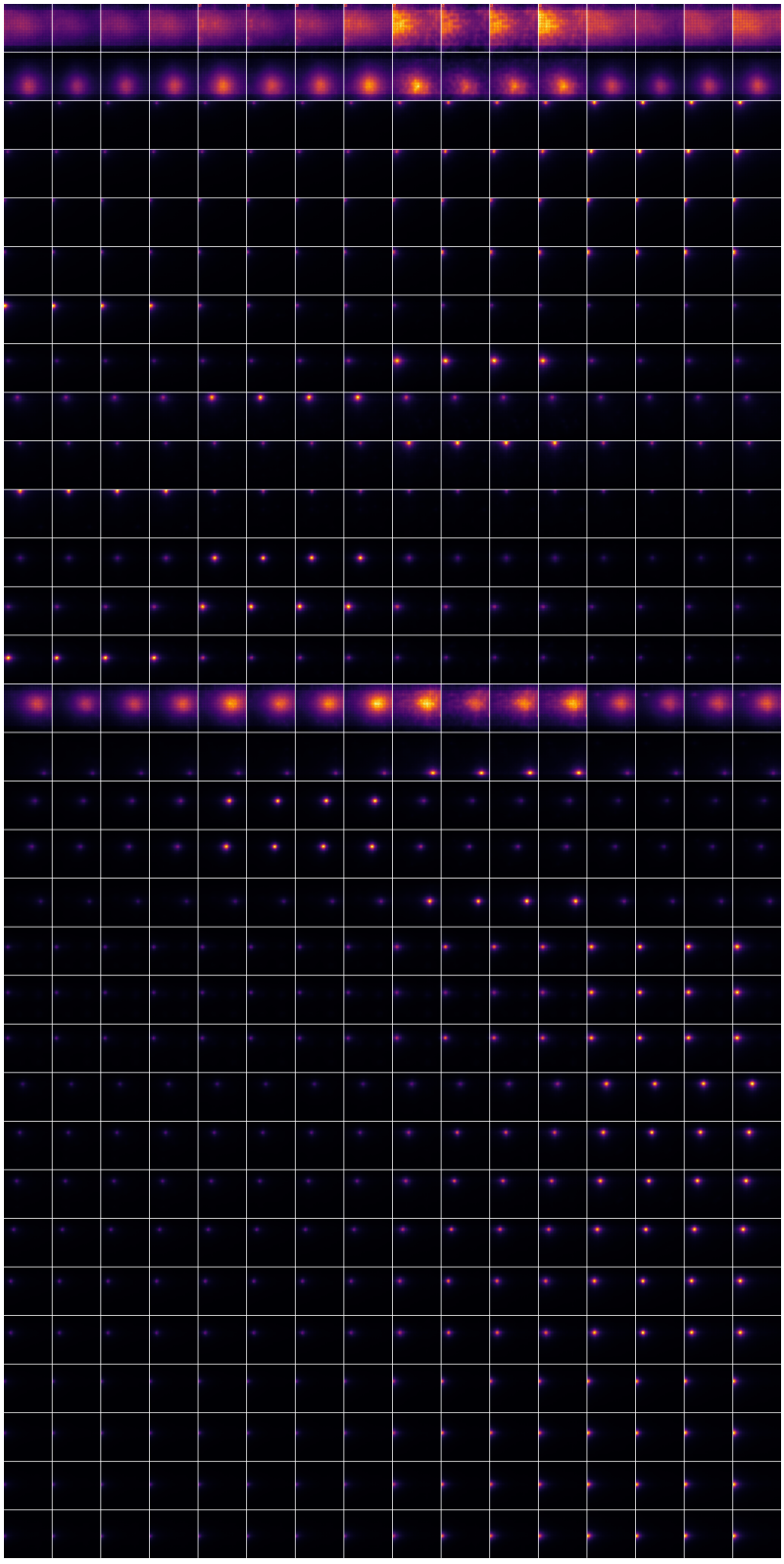


Figure C3: **Average token heatmaps for AdapTok [2].** Every token collapses to one or two sharp, point-like blobs that stay fixed to the same pixels across frames and across videos, i.e. AdapTok's tokens are *positionally* bound like LARP's.

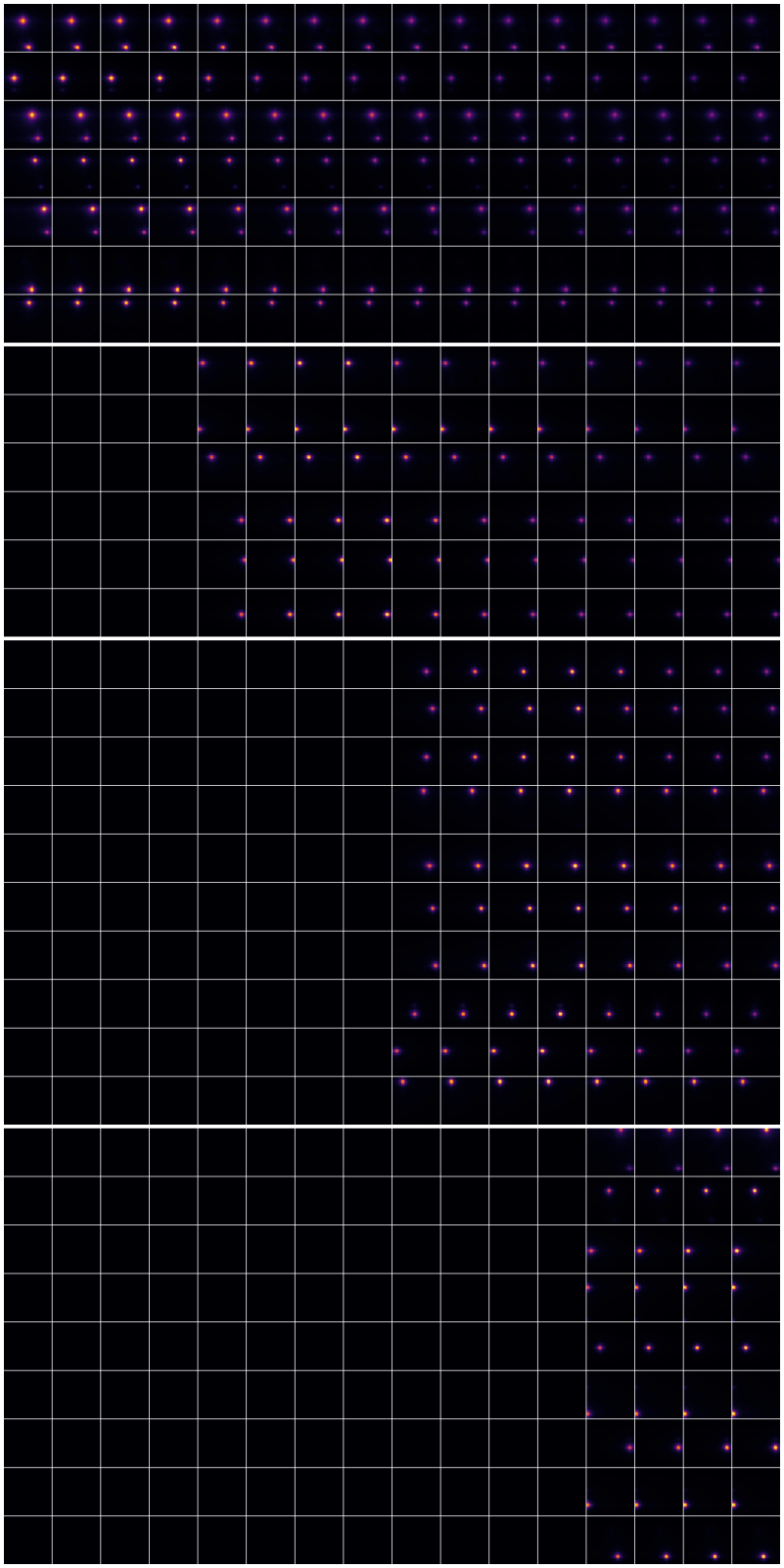


Figure C4: **Average token heatmaps for `TrackTok` without cluster queries.** Without the inductive bias, the encoder reverts to a positional tokenization scheme: each token represents a sharp, spatially-confined blob anchored to a fixed location in the frame, and the tokens jointly tile each frame. Comparing this figure to Fig. C5 isolates the effect of the semantic-clustering bottleneck, which breaks the position-token correspondence and lets token identity track scene content.

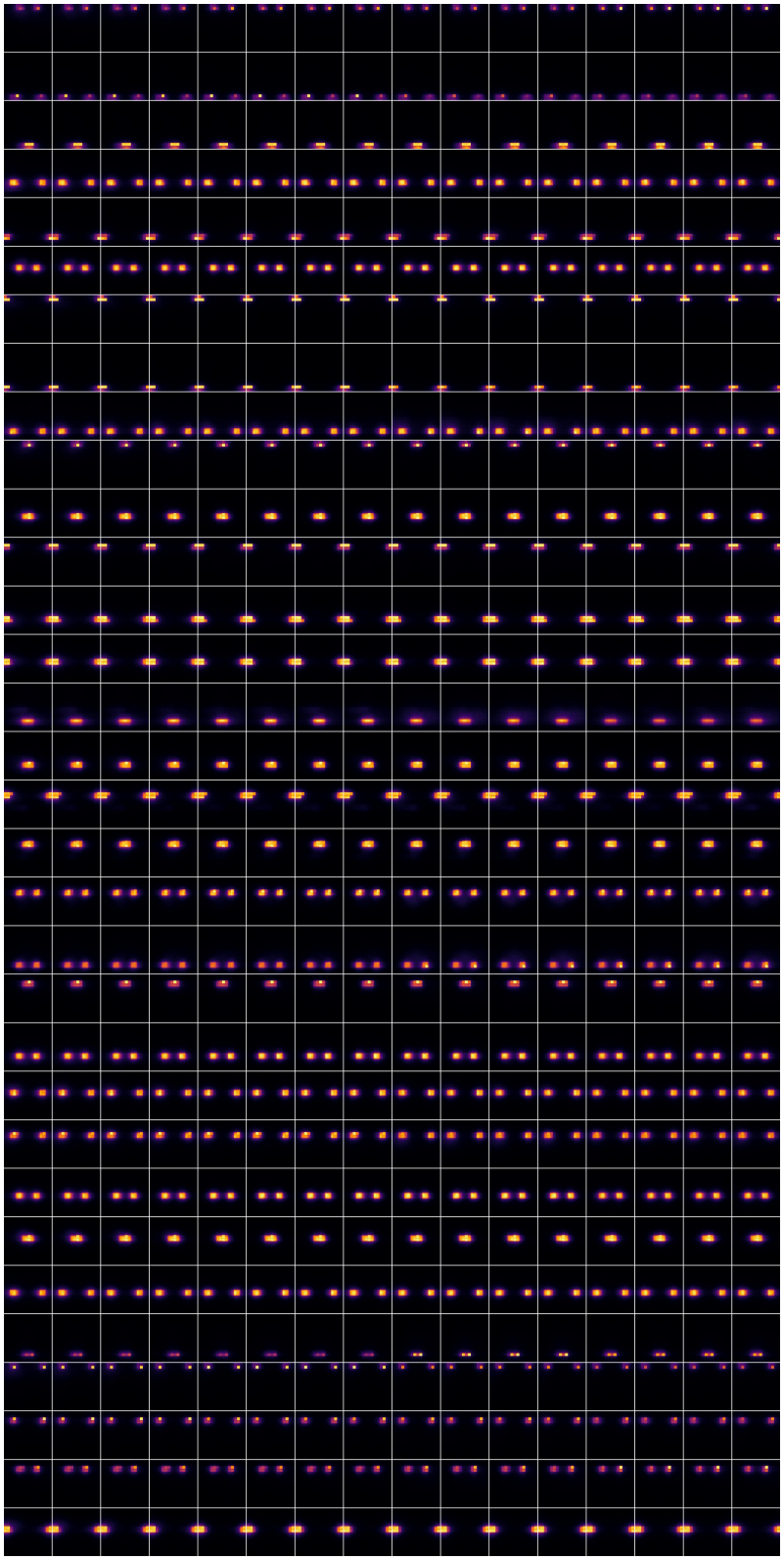
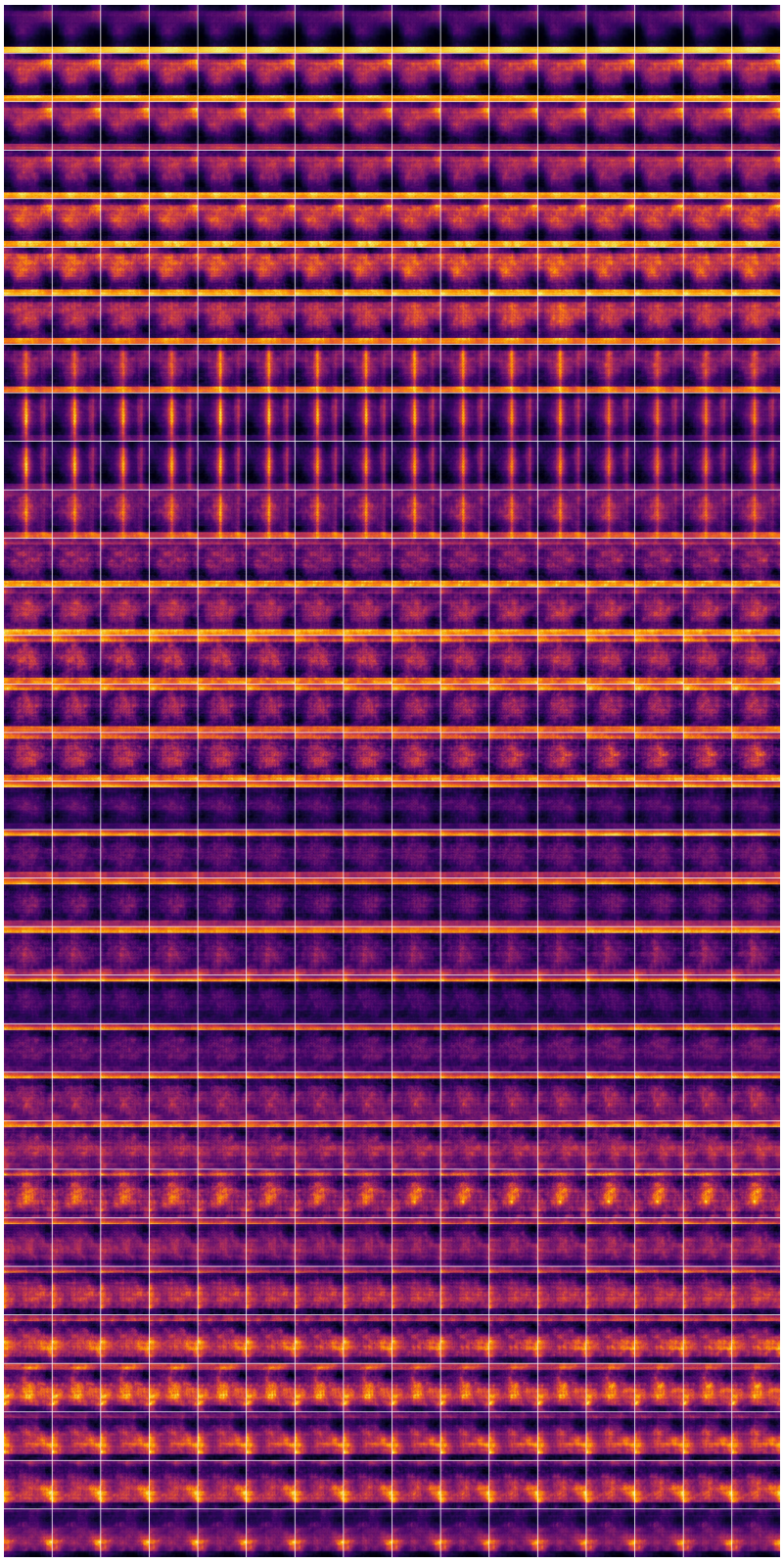


Figure C5: **Average token heatmaps for `TrackTok`.** In contrast to the sharp, position-locked blobs produced by LARP (cf. Fig. C2), `TrackTok`'s content-conditioned tokens yield aggregate maps that are nearly spatially uniform: no token consistently activates on a fixed image region across videos. This is the signature we expect from a *content-allocated* token set: the spatial support of token $\mathbf{z}_i$ depends on the scene rather than on its index, and is consistent with the visualizations in Fig. 6, where each token covers semantically coherent objects whose spatial extent varies between clips.



D Additional Ablations

D.1 Positional Encoding of Cluster Queries

Our design applies k-Means jointly to all V-JEPA 2.1 feature patches $\mathcal{F} = \{\mathbf{f}_i \in \mathbb{R}^C\}_i^F$ of the video clip, where $F = \frac{H}{16} \cdot \frac{W}{16} \cdot \frac{T}{2}$ (Sec. 3.1, paper). A cluster is thus one set of V-JEPA feature patches spanning space and time. For full temporal information, we encode the trajectory, not a single location. First, we take each cluster's per-frame average 2D position (Eq. 1), i.e., the average over the 2D positions of all patches in this frame that belong to the respective cluster. Then in Eq. 2, we bilinearly interpolate frame-wise learnable parameter maps as positional encoding (PE) at these coordinates, concatenate the obtained PEs of all frames (zeros for frames any without patches belonging to the respective cluster), and project them to a single fixed-length trajectory encoding. Each query carries its cluster's full trajectory positional encoding. In Tab. D1, we validate this design choice by providing an additional ablation study versus ours without any PE or with direct PE of a single 3D cluster centroid location using trilinear interpolation of a learnable parameter grid.

Method	#Tokens	#Dim	PSNR↑	LPIPS↓	SSIM↑	rFVD↓	gFVD↓
Ours w/o PE in CQ	128	128	23.32	0.1381	0.7647	68.95	145.91
Ours w/ trilinear PE in CQ	128	128	**23.37**	**0.1371**	**0.7661**	**65.99**	133.02
Ours	128	128	23.36	0.1375	0.7657	66.04	**125.52**

Table D1: Ablation on PE of Cluster Queries

D.2 Representation Autoencoder as Encoder

We evaluate TrackTok with frozen V-JEPA 2.1 [53], VideoMAE [54], and DINOv3 [55] features as alternative encoders. As shown in Tab. D2, our method compresses each encoder's features to 128 tokens of dimension 128, substantially reducing the representation size. While frozen features generally achieve better reconstruction metrics (PSNR, LPIPS, and SSIM), our learned representations achieve competitive reconstruction quality and consistently improve generation quality (gFVD) across all three encoders. This suggests that our tokenization learns compact representations better suited to video generation than directly using frozen features.

Method	#Tokens	#Dim	PSNR↑	LPIPS↓	SSIM↑	rFVD↓	gFVD↓
Frozen V-JEPA2.1	512	1024	**23.99**	**0.1273**	**0.7850**	**47.55**	127.65
Ours w/ V-JEPA2.1	**128**	**128**	23.36	0.1375	0.7657	66.04	**125.52**
Frozen VideoMAE	1568	1024	**34.08**	**0.0284**	**0.9722**	**3.80**	533.60
Ours w/ VideoMAE	**128**	**128**	28.88	0.0636	0.9203	25.28	**220.73**
Frozen DINOv3	1024	1024	**20.92**	**0.1889**	**0.6592**	127.85	339.03
Ours w/ DINOv3	**128**	**128**	20.57	0.2007	0.6451	**126.19**	**204.23**

Table D2: RAE as Encoder Ablation

D.3 Training-Free Token Reduction of V-JEPA 2.1 Features

To assess the benefit of our tokenization design, we compare against simple compression baselines applied to frozen V-JEPA 2.1 features. Specifically, we reduce the original 512-token feature grid to 128 tokens using subsampling and average pooling.

- **Average pooling**: we use kernel $(1, 2, 2)$, averaging each non-overlapping 2×2 spatial neighborhood and producing an $8 \times 4 \times 4$ grid, and
- **Subsampling**: we retain every second feature along both spatial dimensions, producing an $8 \times 4 \times 4$ grid.

Besides average pooling and subsampling, we add a clustering baseline at the same token budget. This provides a strong baseline for testing whether TrackTok's performance gain at low token count could be obtained directly with semantic V-JEPA2.1 cluster centroids as latent tokens, rather than learning a 1D token set from our cluster-initialised queries. Tab. D3 shows that all three direct token-reduction strategies on frozen V-JEPA 2.1 features substantially degrade generation quality, increasing gFVD above 209, whereas TrackTok retains a gFVD of 125.52 at the same 128-token budget. These results show that trivially reducing frozen V-JEPA2.1 features to a smaller token count is insufficient, and thus further support that our TrackTok's cluster queries and tokenizer design are important for maintaining a good generation quality at a low token budget.

Config.	#Tokens	gFVD ↓
Frozen V-JEPA2.1	512 (fixed)	127.65
Frozen V-JEPA2.1 + Average Pooling	128	218.12
Frozen V-JEPA2.1 + Subsampling	128	209.81
Frozen V-JEPA2.1 + Clustering	128	218.30
TrackTok (ours)	128	**125.52**

Table D3: Comparison of tokenization strategies. Lower gFVD is better.

D.4 Number of Clusters K

We vary the number of clusters K at fixed token budgets of 256 and 1024. Intermediate cluster counts yield the best overall results, with $K = 16$ at 256 tokens and $K = 4, 8$ at 1024 tokens providing the strongest trade-off between reconstruction and generation quality. This suggests that both too few and too many clusters can impair the allocation of representational capacity. In addition, we observe that the optimal number of clusters K varies with the token budget, which suggests that a fixed K is not optimal across settings. Developing an adaptive strategy to allocate tokens across clusters remains an interesting direction for future work.

Method	#Tokens	#Dim	K	PSNR↑	LPIPS↓	SSIM↑	rFVD↓	gFVD↓
Ours	1024	16	1	21.81	0.1919	0.6743	200.12	386.38
Ours	1024	16	2	22.57	0.1726	0.7049	135.22	318.87
Ours	1024	16	4	<u>23.79</u>	<u>0.1292</u>	<u>0.7795</u>	<u>62.79</u>	**116.50**
Ours	1024	16	8	**23.85**	**0.1280**	**0.7814**	**59.31**	<u>124.11</u>
Ours	1024	16	16	23.16	0.1429	0.7569	69.22	226.26
Ours	1024	16	32	22.57	0.1552	0.7354	84.90	209.18
Ours	1024	16	64	23.00	0.1465	0.7506	70.71	261.17

Table D4: Ablation on #Clusters K at 1024 tokens

Method	#Tokens	#Dim	K	PSNR↑	LPIPS↓	SSIM↑	rFVD↓	gFVD↓
Ours	256	64	1	23.28	0.1550	0.7320	125.12	259.90
Ours	256	64	16	**24.17**	**0.1391**	**0.7621**	<u>64.16</u>	128.73
Ours	256	64	32	23.85	0.1454	0.7513	67.93	130.46
Ours	256	64	64	<u>24.05</u>	<u>0.1418</u>	<u>0.7577</u>	**62.72**	131.01

Table D5: Ablation on #Clusters K at 256 tokens

D.5 Token Dimension

We ablate the token dimension while fixing the token budget at 128. Increasing the dimension consistently improves both reconstruction and generation quality, with ($d = 128$) achieving the best results across metrics, highlighting the importance of sufficient per-token representational capacity.

Method	#Tokens	#Dim	PSNR↑	LPIPS↓	SSIM↑	rFVD↓	gFVD↓
Ours	128	128	**23.36**	**0.1375**	**0.7657**	**66.04**	**125.52**
Ours	128	64	22.90	0.1469	0.7493	79.25	146.78
Ours	128	32	22.35	0.1592	0.7282	103.03	184.45
Ours	128	**16**	21.88	0.1710	0.7077	134.77	250.48

Table D6: Ablation on Token Dimension

D.6 Text-Based Semantic Baseline

To evaluate whether language-level semantic representations suffice for video reconstruction, we caption 16-frame UCF-101 clips using pretrained Qwen2.5-VL [56] and reconstruct the videos from captions alone using Wan2.1-T2V-1.3B [8]. We generate four candidates per caption and report both the best candidate selected by LPIPS (oracle-4) and a fixed single-candidate result. We evaluate both using the same protocol as TrackTok on UCF-101. As shown in Tab. D7, despite capturing high-level semantics, captions alone yield substantially poorer reconstruction quality. This suggests that language-level descriptions lack the spatial grounding and fine-grained appearance, motion, and scene-specific details needed for faithful video reconstruction.

Method	Representation	rFVD ↓	LPIPS ↓
Qwen2.5-VL + Wan2.1-T2V-1.3B (oracle-4)	Video caption	3033.90	0.7999
Qwen2.5-VL + Wan2.1-T2V-1.3B	Video caption	3097.57	0.8459
Ours (128 tokens)	Continuous tokens	**63.89**	**0.1495**

Table D7: **Text-Based Baseline.** Using a video-captioning model (Qwen2.5-VL) together with a text-to-video generator (Wan2.1-T2V-1.3B) as a video autoencoder fails to faithfully reconstruct the input video, in contrast to TrackTok.

E Qualitative Examples

In this section we include additional qualitative examples for unsupervised object tracking, per-sample video reconstruction and token visualization, as well as generation results.

Tracking. Fig. E1, E2, and E3 show three YouTube-VIS 2021 validation videos, with 8 columns spread uniformly over the whole video (up to 62 frames, i.e. up to 16 consecutive 4-frame clips). The *top row* shows the input frames with the ground-truth annotations, and each further row shows the object-level token clusters of each tokenizer obtained by linking the 28 tokens of each clip across clips and merging them into tracks (also quantitatively reported in Table 2).

Reconstruction & Token Visualization. In Fig. E4, E5, and E6, each consists of three rows: *input frames (top)*, *reconstructed frames with `TrackTok` (middle)*, and the *token visualization*, with 8 columns showing every other frame of the $16\times128\times128$ clip. The clips shown for each token budget are sampled uniformly at random from the UCF-101 validation split with a fixed seed of 42, uncurated so the examples reflect the typical (rather than best-case) behavior of the tokenizer. Across the 32-, 128-, and 256-token configurations the reconstructions become progressively sharper as the token budget grows, while the PCA overlay consistently shows clusters that follow the underlying scene content across frames rather than staying locked to fixed image locations – the same content-allocated tokenization behavior that the aggregate analysis of Sec. C characterizes at the dataset level.

Generation. Fig. E7, E8,and E9 show $16\times128\times128$ clips sampled from LightningDiT trained on `TrackTok` latents at 32, 128, and 256 tokens, for randomly selected UCF-101 action classes. As with the reconstructions, the samples become sharper and more temporally coherent as the token budget grows, while even the 32-token model already produces plausible motion and scene layout.

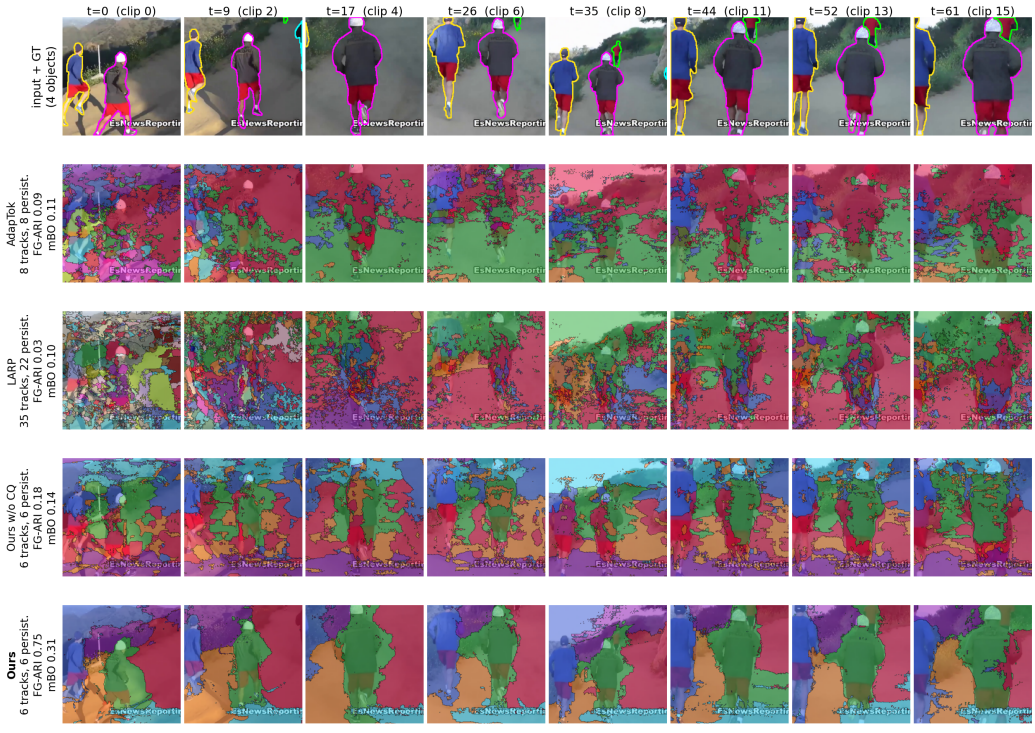


Figure E1: **Unsupervised object tracking on a YouTube-VIS 2021 validation video** (62 frames, 4 objects), one fixed colour per track. `TrackTok` keeps each runner in a single track for all frames, while the baselines fragment objects into many short-lived tracks and our `TrackTok` without cluster queries (CQ) binds tracks to image regions that are not semantically-aligned.

Figure E2: `TrackTok` mostly keeps each object in one track for all frames, while the baselines fragment objects into many short-lived tracks.

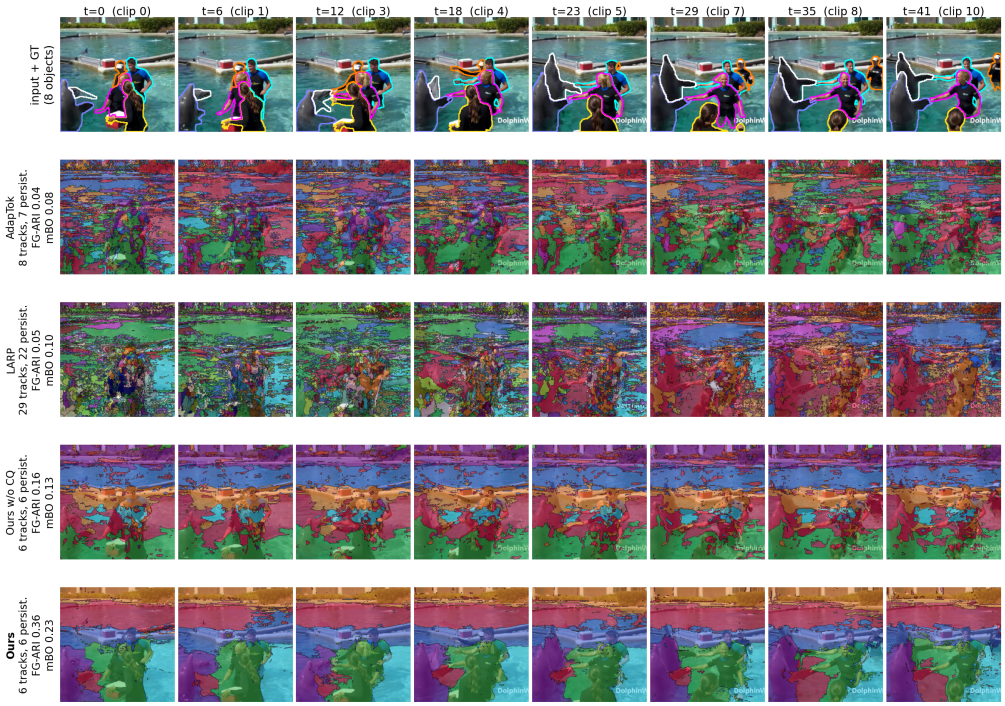


Figure E3: `TrackTok` mostly keeps each object in one track for all frames, while the baselines fragment objects into many short-lived tracks.

Figure E4: **Random qualitative reconstruction and token impact from `TrackTok` at 32 tokens.**

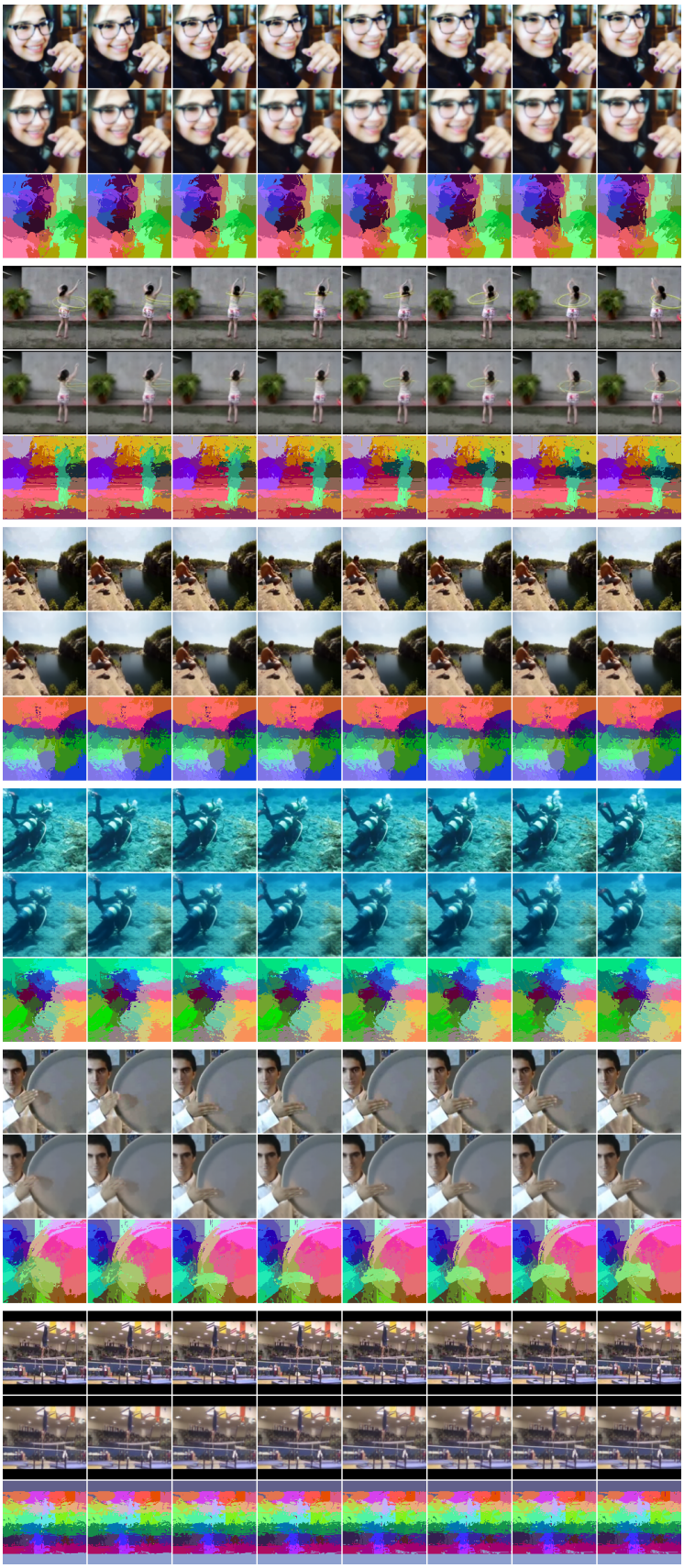


Figure E5: **Random qualitative reconstruction and token impact from `TrackTok` at 128 tokens**

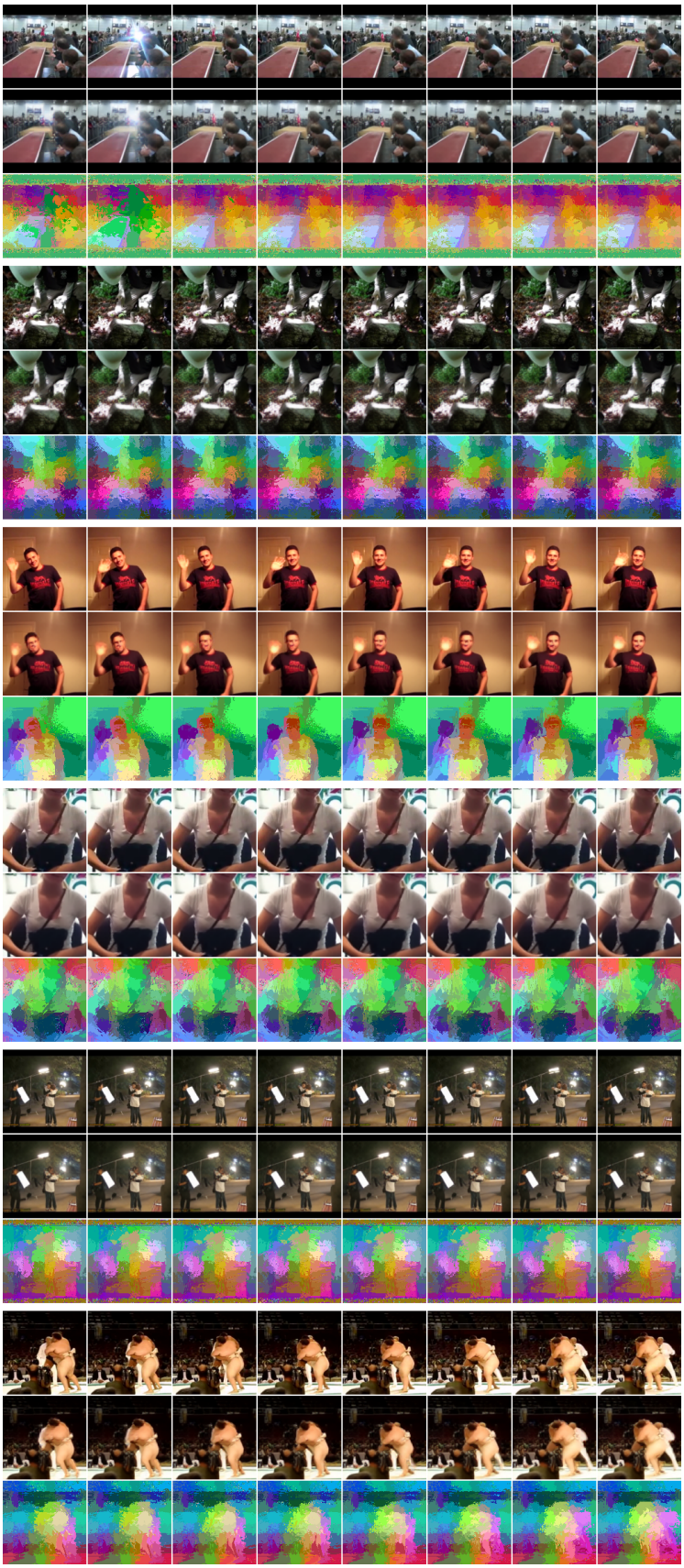


Figure E6: **Random qualitative reconstruction and token impact from `TrackTok` at 256 tokens.**

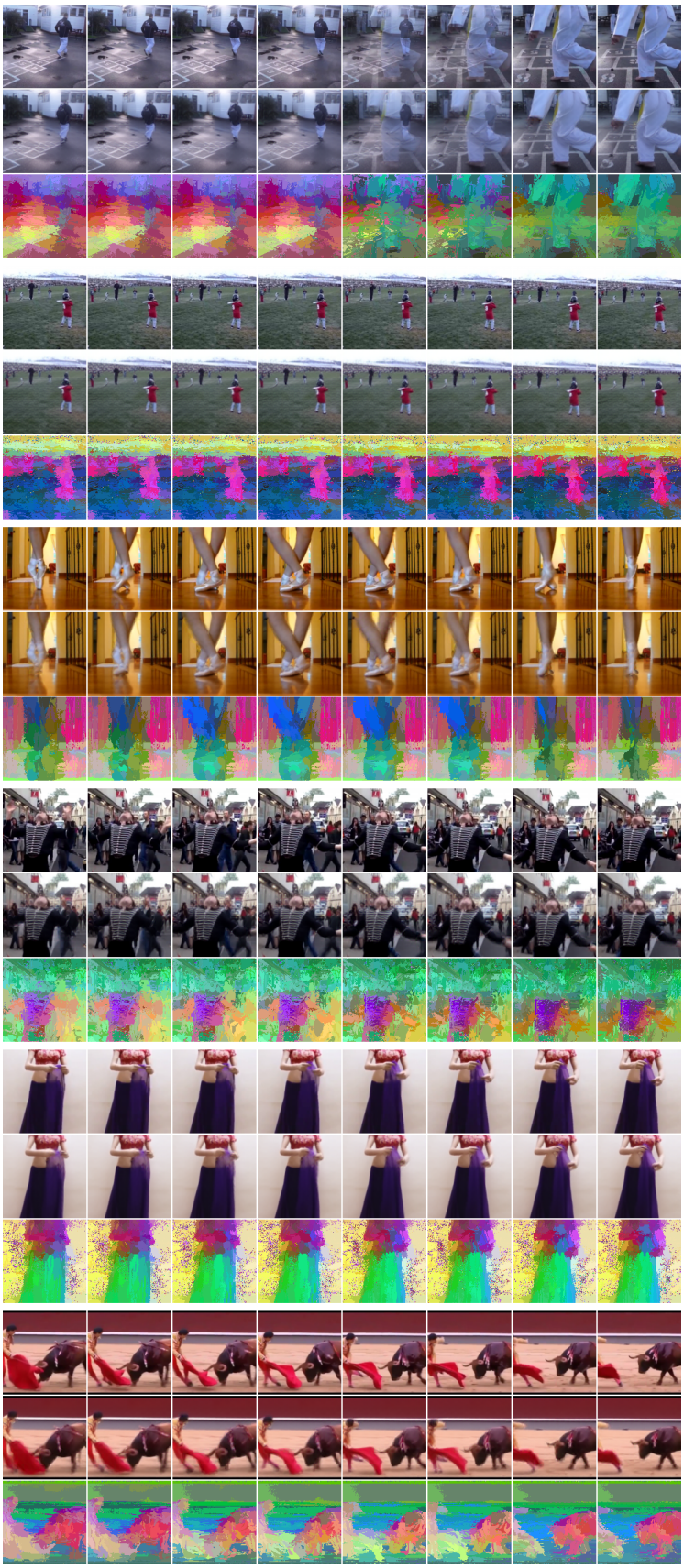


Figure E7: **Generated video clips of $16 \times 128 \times 128$ resolution across randomly selected action classes in UCF-101, using LightningDiT trained on `TrackTok` latents at 32 tokens.**

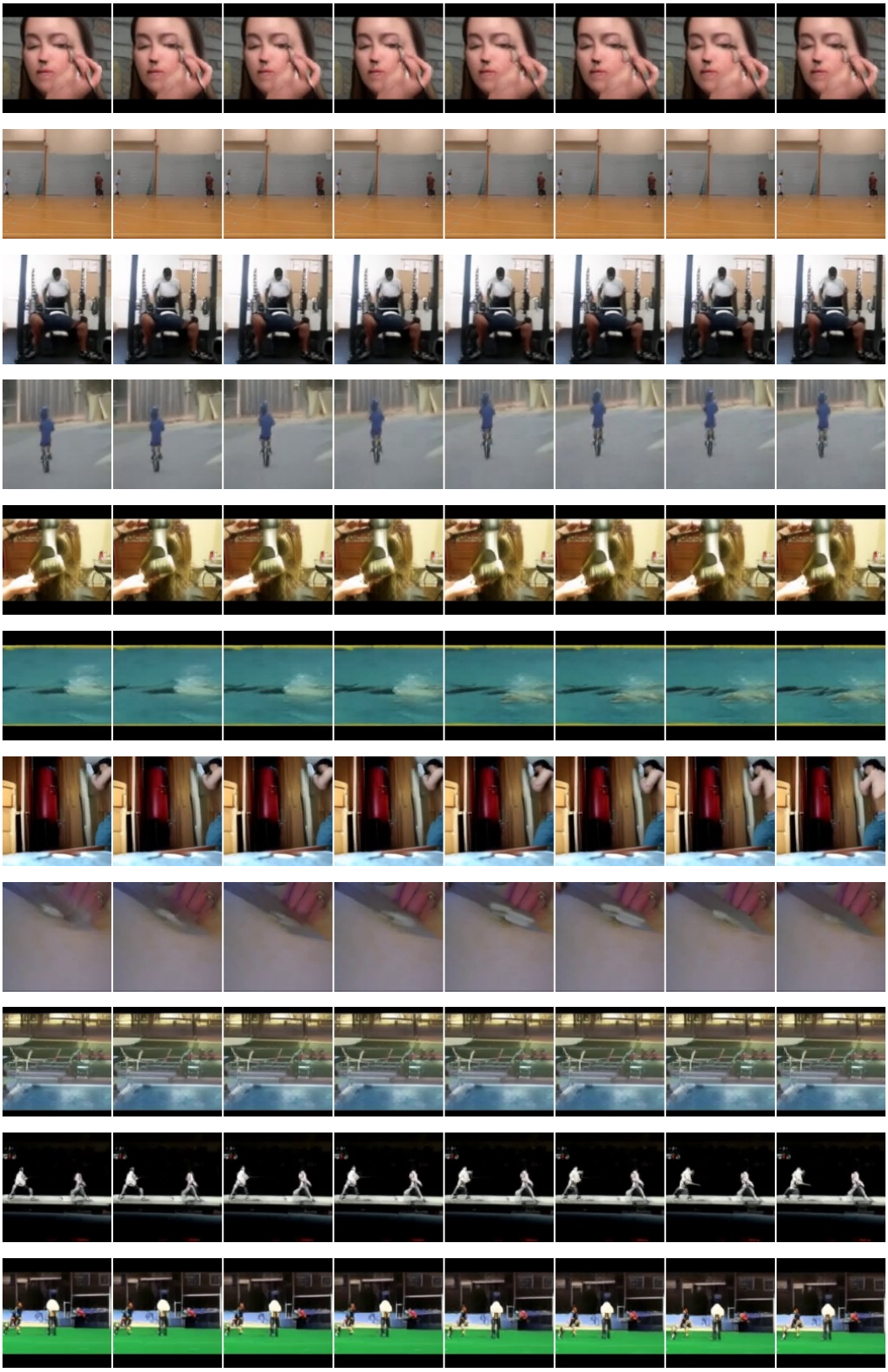


Figure E8: **Generated video clips of** $16 \times 128 \times 128$ **resolution across randomly selected action classes in UCF-101, using LightningDiT trained on** `TrackTok` **latents at** 128 **tokens.**

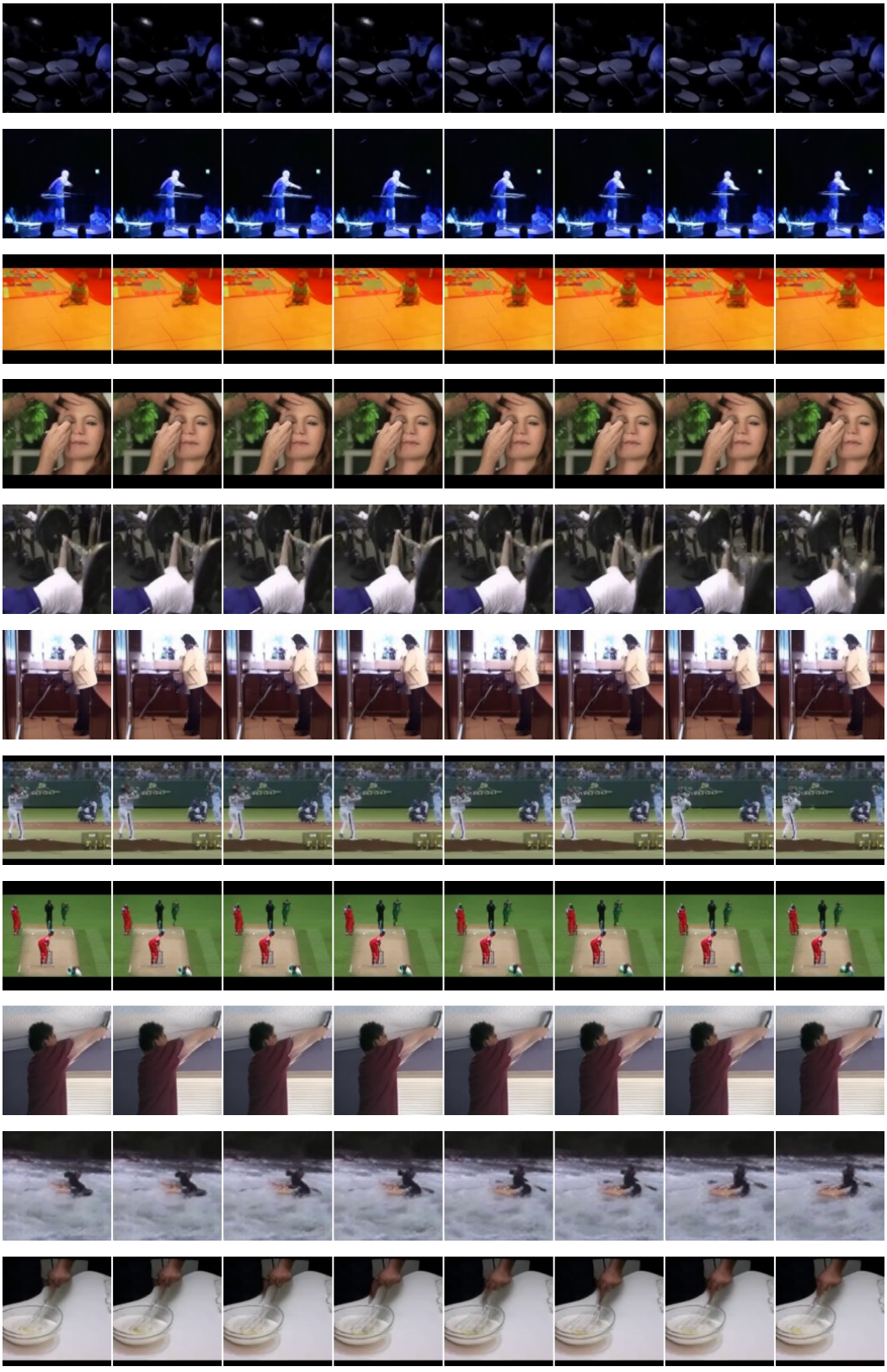
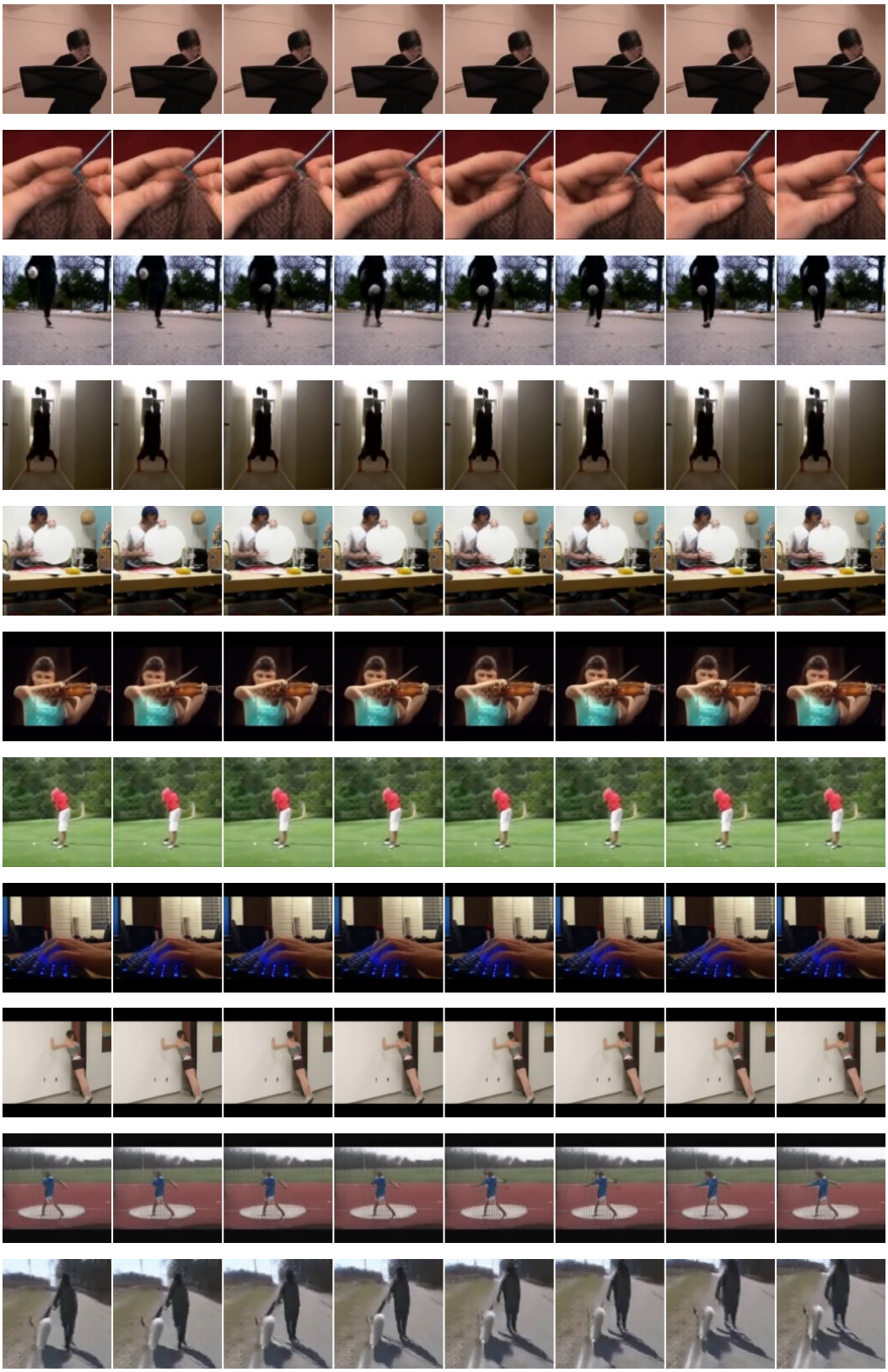



Figure E9: **Generated video clips of $16 \times 128 \times 128$ resolution across randomly selected action classes in UCF-101, using LightningDiT trained on `TrackTok` latents at 256 tokens.**