

# Recurrent Video Masked Autoencoders

## Supplementary Material

### 7. Training data details

We use a data mixture very similar to the one proposed in [3], consisting of only data from publically available video datasets. However, we do not apply any extra curation to these datasets and critically don't rely on ImageNet for additional image-level data as so many prior works do:

Source	Samples	Type	FPS	Apply Curation	Weight
SSv2 [29]	168K	EgoVideo	25	No	0.056
Kinetics-700 [14]	733K	ExoVideo	25	No	0.188
Howto100M [50]	1.1M	ExoVideo	10	No	0.318
YT8M [1]	3.3M	ExoVideo	10	No	0.188
YT-BoundingBoxes [63]	380K	ExoVideo	10	No	0.250

Table 4. Dataset usage and statistics. We use only video datasets. While we apply no curation ourselves to any of these datasets, the original dataset construction for many of these datasets (except YT8M) did involve significant curation. For YT8M we utilize available clips- a significant number of clips from the original dataset can no longer be accessed.

A training batch consists of selecting videos with the mixture weights specified in Table 4. Each clip from a given datasets is a 64 frame clip (which corresponds to different durations because of the differing fps for each source). We use the first 4 consecutive frames for the source frames and sample a target frame with a uniform temporal gap of 4 to 48 frames. For each video clip we apply the following augmentations:

1. Video-level RandomHorizontalFlipping ( $p = 0.5$ )
2. Frame-level RandomResizedCrop with scale = (0.3, 1.0) and aspect ratio = (0.75, 1.25), using bicubic interpolation.

### 8. Architecture details

We provide the network architecture details for each model component in Tables 5 and 6. As specified in the main text,

Table 5. RVM Architecture Variants. We scale the Encoder and RNN core across four sizes (S, B, L, H). The Encoder follows standard ViT specifications [24]. The RNN core dimension matches the encoder embedding dimension.

Model	ViT Encoder				RNN Core		Total Params
	Embed Dim	Heads	Layers	MLP Ratio	Layers	Heads	
RVM-S	384	6	12	4.0	4	8	34M
RVM-B	768	12	12	4.0	4	12	117M
RVM-L	1024	16	24	4.0	4	16	375M
RVM-H	1280	16	32	4.0	4	16	743M

RVM-S,B,L models are trained for 1M steps (approx. 2B samples). However, we find that larger models do benefit from even longer schedules and thus train our RVM-H for 4B steps.

Table 6. Decoder Architecture. The decoder is fixed across all model sizes. Each block consists of Cross-Attention (Target-Source), MLP, and Self-Attention layers. Refer to pseudocode in Section 10

Hyperparameter	Value
Embedding Dimension ( $D_{dec}$ )	512
Number of Heads	16
Number of Blocks	8
MLP Ratio	4.0
<i>Block Structure</i>	
1. Cross-Attention	Target (Q) $\leftrightarrow$ Source (K,V)
2. Feed-Forward	MLP
3. Self-Attention	Target (Q,K,V)

### 9. Self attention ablation details

To ensure a fair comparison between our recurrent temporal aggregation and a full self-attention approach, we minimized differences in the experimental setup. We maintained the exact same encoder, architecture, and hyperparameters. The primary distinction lies in how tokens are prepared for the ViT. In the full self-attention baseline, we patchify and project frames independently but concatenate all resulting tokens along the token axis before feeding them into the ViT (accounting for the extra layers due to the RNN core.). We also augmented the positional embeddings with a time dimension to provide temporal context. This setup is essentially equivalent to using a  $1 \times 16 \times 16$  patch size in spatiotemporal models like VideoMAE. All other components, including masking patterns, training objectives, and learning schedules, remained identical to the RNN configuration.

### 10. Pseduocode

```

1 class RVMCell(nn.Module):
2     def __init__(self, dim,
3         transformer_block):
4         super().__init__()
5         self.Tx = transformer_block
6         # Update (u) and Reset (r) gate
7         self.We_u, self.Ws_u = nn.Linear(
8             dim, dim), nn.Linear(dim, dim)
9         self.We_r, self.Ws_r = nn.Linear(
10            dim, dim), nn.Linear(dim, dim)

11     def forward(self, x_seq):
12         """

```

```

951 11     x_seq: Sequence of source frame
952     tokens [e_1, ..., e_K]
953 12     Returns: Sequence of refined
954     features [o_1, ..., o_K]
955 13     """
956 14     # Initialize state s_0 to zero
957 15     s = torch.zeros_like(x_seq[0])
958 16     outputs = []
959
960 17     for x in x_seq:
961 18         # 1. Compute Gates (Eq. 1)
962 19         u = torch.sigmoid(self.We_u(x)
963 + self.Ws_u(s))
964 21         r = torch.sigmoid(self.We_r(x)
965 + self.Ws_r(s))
966 22
967 23         # 2. Transformer Integration (
968 Eq. 2)
969 24         # Query is current input (x);
970 KV is reset-gated state
971 25         h = self.Tx(query=x, kv=r * s)
972 26
973 27         # 3. State Update (Eq. 2)
974 28         s = (1 - u) * s + u * h
975 29
976 30         # 4. Output (Eq. 3)
977 31         outputs.append(s)
978 32
979 33     return torch.stack(outputs)

```

Listing 1. RVM Recurrent Core Pseudo-code

```

980 1 class TransformerBlock(nn.Module):
981 2     def __init__(self, dim, num_heads):
982 3         super().__init__()
983 4         self.ln1 = nn.LayerNorm(dim)
984 5         self.self_attn = nn.
985 MultiheadAttention(dim, num_heads)
986 6
987 7         self.ln2 = nn.LayerNorm(dim)
988 8         self.cross_attn = nn.
989 MultiheadAttention(dim, num_heads)
990 9
991 10        self.ln3 = nn.LayerNorm(dim)
992 11        self.mlp = MLP(dim) # Standard Feed
993 Forward
994
995 12    def forward(self, x, mem):
996 13        """
997 14        x: Current frame tokens (Query) [
998 cite: 350]
999 16        mem: Gated previous state (Key/
1000 Value) [cite: 350, 476]
1001 17        """
1002 18        # 1. Self-Attention (Intra-frame
1003 mixing)
1004 19        # Using pre-normalization [cite:
1005 341]

```

```

20         x = x + self.self_attn(query=self.
1006 ln1(x),
1007
21                                     key=self.ln1
1008 (x),
1009
22                                     value=self.
1010 ln1(x))[0]
1011
23
1012 # 2. Feed Forward
1013 x = x + self.mlp(self.ln3(x))
1014
25
1015 # 3. Cross-Attention (Temporal
1016 integration)
1017 # Queries from current frame, Keys/
1018 Values from history
1019 x = x + self.cross_attn(query=self.
1020 ln2(x),
1021
29                                     key=mem,
1022
30                                     value=mem)
1023
31 [0]
1024
32
1025
33     return x
1026

```

Listing 2. Cross Attention block used in RNN core and decoder

## 11. Evaluation Details

To comprehensively assess the capabilities of Recurrent Video Masked Autoencoders (RVM), we evaluate the model across a broad spectrum of 8 diverse datasets covering high-level semantics, low-level geometry, and temporal correspondence. Our evaluation suite encompasses distinct visual tasks including action recognition (SSv2 [30], Kinetics-700 [42]), monocular depth estimation (ScanNet [21]), and fine-grained motion tracking (Perception Test [56], Waymo Open [66]). Additionally, we probe the spatio-temporal consistency of the learned features through non-parametric nearest-neighbor label propagation on the DAVIS-2017 [59], JHMDB [40], and VIP [80] benchmarks. This exhaustive protocol ensures a holistic comparison against existing state-of-the-art video and image foundation models.

### 11.1. Downstream tasks

We adopt the rigorous evaluation protocol of Carreira et al. [15], attaching lightweight attention-based readouts to frozen backbones.

- **SSv2 action recognition** [30]: A fine-grained dataset requiring temporal understanding. We process 16-frame clips at  $224 \times 224$  resolution with a stride of 2. The readout employs a cross-attention layer with 768 channels and 12 heads, using a single learned query to pool representations before the final linear classifier. Training involves color augmentation (brightness, contrast, saturation, hue) and random grayscale conversion. We report top-1 accuracy (%).
- **Kinetics-700-2020 action recognition** [42]: A large-scale benchmark for broad action understanding. Similar to

Table 7. **Downstream Task Readout Hyperparameters.** Summary of the attention-based readout configurations used for each task, following the protocol of Carreira et al. [15]. All readouts use a Cross-Attention (CA) mechanism on top of the frozen backbone features.

Task	Input Shape	Readout Arch	Heads / Channels	Query Type
SSv2	$16 \times 224^2$	Cross-Attn	12 / 768	Learned Vector
Kinetics-700	$16 \times 224^2$	Cross-Attn	16 / 1024	Learned Vector
ScanNet	$16 \times 224^2$	Cross-Attn	16 / 1024	Learned Vector
Perception Test	$16 \times 224^2$	Cross-Attn	8 / 1024	Point Coord. + Fourier
Waymo Open	$16 \times 256^2$	Cross-Attn	4 / 1024	Box Coord. + Fourier

SSv2, we use 16-frame clips with a stride of 2. The readout is larger, utilizing 1024 channels and 16 heads with a single learned query. For evaluation, we average predictions over 7 linearly spaced temporal clips per video. We report top-1 accuracy (%).

- **ScanNet depth estimation [21]:** Evaluates geometric understanding on indoor RGB-D videos. We input 16 RGB frames and predict dense depth maps. The readout uses cross-attention (1024 channels, 16 heads) where queries are learned features corresponding to each  $2 \times 8 \times 8$  patch. The model minimizes an  $L_2$  loss on log-scale depth. Performance is measured by Absolute Relative Error (AbsRel).
- **Perception Test point tracking [56]:** Measures fine-grained long-term motion tracking. The readout uses cross-attention (1024 channels, 8 heads) where queries are derived from the initial point positions embedded via Fourier features. The model predicts position, visibility, and uncertainty for each track. Following Carreira et al. [15], the readout is trained on the synthetic Kubric MOVI-E dataset [31] before evaluating on the real-world Perception Test. We report Average Jaccard (AJ).
- **Waymo Open object tracking [66]:** Assesses object-level motion consistency in driving scenarios. We track 2D bounding boxes over 16-frame clips ( $256 \times 256$  resolution). The readout employs cross-attention (1024 channels, 4 heads) with queries formed from the initial bounding box coordinates. We report mean Intersection-over-Union (mIoU).

## 11.2. Nearest-neighbor tasks

Unlike read-out classification, this protocol directly probes whether the pre-trained features encode spatially and temporally consistent information without any task-specific training.

- **DAVIS-2017 video segmentation tracking [59]:** A video object segmentation benchmark with diverse object categories and complex motion. The task is to propagate ground-truth instance masks provided in the first frame across subsequent frames. We adopt the non-parametric label propagation algorithm of Jabri et al. [38] that considers the similarity between patch features across frames, using

Table 8. **Label Propagation Evaluation Protocols.** Summary of hyperparameters used across DAVIS, JHMDB, and VIP tasks. The models share the same temperature and memory bank size, differing mainly in resolution and  $k$ -NN retrieval count.

Parameter	DAVIS-2017	JHMDB	VIP
Task	VOS	Keypoint Tracking	Part Propagation
Resolution	$480 \times 880$	$320 \times 320$	$448 \times 880$
Metric	$\mathcal{J} \& \mathcal{F}$ Mean	PCK@0.1	mIoU
Top- $k$ ( $k$ )	7	7	10
Algorithm	Non-parametric Label Propagation [38]		
Temperature ( $\tau$ )		0.7	
Context Frames		20	
Search Radius		20	

480p resolution with patch sizes 14/16 matched across models. Like DINO [13], performance is reported in the standard  $\mathcal{J} \& \mathcal{F}$ -mean metric, which combines region similarity ( $\mathcal{J}$ ) and contour accuracy ( $\mathcal{F}$ ) [57], computed at the native resolution of the videos.

- **JHMDB human keypoint tracking [40]:** A dataset of short video clips for human pose estimation and action understanding. We follow the setup of Li et al. [46], using  $320 \times 320$  video resolution and a single context frame, and report PCK@0.1.
- **VIP human part tracking [75]:** A video instance segmentation benchmark requiring pixel-level separation of multiple moving instances. [80] requires dense propagation of semantic part masks across long human-centric videos, with up to 20 different human part categories and durations of 120 seconds. Following the protocol of Li et al. [46], we evaluate at  $448 \times 880$  resolution using a single context frame.

## 12. Baseline Models

**Backbone architectures** We evaluate models including SiamMAE, DINOv2, VideoMAE, VideoMAEv2, V-JEPA, and 4DS. Most baselines use Vision Transformers (ViTs) with spatio-temporal patch tokenization of size (2, 16, 16), where each token covers two consecutive frames and a  $16 \times 16$  spatial region. Self-attention is applied across all tokens, making computation quadratic in the number of patches. We evaluate models across a wide range of capacities, from ViT-S ( $\sim 30$ M parameters) up to ViT-H ( $\sim 700$ M parameters). The exact configurations, pre-training checkpoints, and architectural details for each model are provided in Table 9. Below, we provide a concrete description of each model included in our experiments.

### 12.1. VideoMAE and VideoMAEv2

As a representative video-masked autoencoder, VideoMAE [27, 36] operates on a standard Vision Transformer (ViT) backbone processing tubelets of size  $2 \times 16 \times 16$ . It employs a high masking ratio and reconstructs normalized

pixels of masked regions using a vanilla ViT decoder. Building on this, VideoMAE v2 [68] incorporates a dual masking strategy—masking tokens in both the encoder and decoder—to enhance computational efficiency and scalability. We examine variants ranging from ViT-B (30M parameters) to the billion-parameter ViT-g. These models are typically pretrained on Kinetics-400, with v2 leveraging a progressive training schedule on a massive mixed dataset of public videos. We utilize the official checkpoints respectively<sup>1,2</sup>.

## 12.2. V-JEPA

The V-JEPA family utilizes a Joint-Embedding Predictive Architecture (JEPA) to learn semantic video representations. A ViT encoder [24] processes 16-frame inputs (resolution  $224 \times 224$ ) decomposed into  $2 \times 16 \times 16$  patches. Unlike generative approaches, V-JEPA is trained to predict the latent representation of a target video signal  $y$  from a context  $x$  (a heavily masked version of  $y$ ) by minimizing the  $L_1$  distance in feature space. The target encoder is updated via an exponential moving average (EMA) of the context encoder. We evaluate ViT-L ( $\sim 300$ M) and ViT-H ( $\sim 600$ M) variants pretrained for 90k iterations on VideoMix2M, a compilation of HowTo100M, Kinetics-400/600/700 (K710) [41], and Something-Something-v2 [29]. We use the official model checkpoints<sup>3</sup>.

## 12.3. DINOv2

DINOv2 [52] adapts the DINO self-distillation framework to large-scale data. It processes video frames independently as images using a ViT [24] with patch size 14, yielding a feature grid of  $16 \times 16 \times 16$  for a 16-frame input. The training objective combines contrastive and distillation losses at both the image and patch levels, supported by sophisticated data curation and regularization techniques. We utilize the official pre-trained checkpoints<sup>4</sup> for ViT-L (307M) and ViT-g (1.1B), which were trained for 625k steps with a batch size of 3,072, applying the model frame-by-frame to generate video features.

## 12.4. 4DS

The 4DS framework [15] simplifies the masked autoencoding paradigm (SimpleMAE) by discarding the separate lightweight decoder in favor of using the last few self-attention blocks of the encoder for reconstruction. It employs a standard ViT with  $2 \times 16 \times 16$  tokenization but opts for a random masking strategy (95% ratio) over tube masking. The model minimizes the  $L_2$  reconstruction loss on RGB values across all patches—both masked and unmasked—without target normalization. We evaluate a ViT-B

variant pretrained on a massive corpus of 170 million web videos (1 billion clips). We use the official checkpoints<sup>5</sup>.

Table 9. **Summary of Pre-trained Models used for Nearest Neighbor Evaluation.** We report the architecture, patch size ( $P$ ), embedding dimension ( $D$ ), depth ( $L$ ), number of heads ( $H$ ), and the pre-training dataset for each checkpoint used.

Model	Arch.	$P$	$D$	$L$	$H$
VideoMAE	ViT-L	$16 \times 16$	1024	24	16
VideoMAE v2	ViT-L	$16 \times 16$	1024	24	16
V-JEPA	ViT-L	$16 \times 16$	1024	24	16
DINOv2	ViT-L	$14 \times 14$	1024	24	16
4DS-VideoMAE	ViT-B	$16 \times 16$	768	12	12
RVM	ViT-L	$16 \times 16$	1024	24	16

## 13. Additional Qualitative Results

To probe the spatiotemporal structure of the learned representations, we visualize the dense feature maps extracted from the frozen backbone of each model. We employ two standard dimensionality reduction techniques:

**Principal Component Analysis (PCA).** We compute the top-3 principal components of the flattened feature tokens across the entire video volume. These components are whitened and mapped to RGB color channels. This visualization highlights the global structure and smoothness of the feature space, revealing whether the model separates foreground motion from the background.

**K-Means Clustering.** We apply K-means clustering with  $k = 5$  clusters (initialized via k-means++) to the feature descriptors. Each cluster is assigned a distinct color to generate a segmentation mask. This acts as a proxy for semantic understanding, testing whether spatially coherent regions (e.g., an object’s parts) are grouped together and whether these assignments remain temporally consistent across frames. As shown in Figures 8–11, RVM demonstrates remarkable temporal consistency. In dynamic sequences like *car-roundabout* and *pigs*, RVM maintains stable cluster assignments for moving objects, resisting the “flickering” artifacts observed in VideoMAE and VideoMAE v2. While DINOv2 produces high-quality semantic segments, it lacks temporal awareness; RVM matches this semantic stability while explicitly modeling the temporal evolution of the instance.

**Video Object Segmentation (DAVIS-2017).** We visualize the quality of spatiotemporal feature correspondences

<sup>1</sup><https://github.com/MCG-NJU/VideoMAE>

<sup>2</sup><https://github.com/OpenGVLab/VideoMAEv2>

<sup>3</sup><https://github.com/facebookresearch/jepa>

<sup>4</sup><https://github.com/facebookresearch/dinov2>

<sup>5</sup><https://github.com/google-deepmind/representations4d>



through non-parametric label propagation on the DAVIS-2017 validation set. Using a context queue of 20 frames and  $k = 7$  nearest neighbors, RVM demonstrates robust object segmentation capabilities. By leveraging the recurrent memory, the model effectively propagates ground-truth masks from the initial frame to subsequent timesteps. The learned representations exhibit strong temporal stability, maintaining precise object boundaries even in the presence of fast motion and partial occlusions.

**Human Pose Tracking (JHMDB).** To assess fine-grained motion understanding, we evaluate keypoint tracking on the JHMDB dataset. We propagate human joint annotations using the same protocol as DAVIS ( $k = 7$ ). RVM captures the structural articulation of the human body, tracking individual keypoints (e.g., wrists, elbows, knees) with high precision. The model’s recurrent mechanism ensures that feature trajectories remain consistent over time, minimizing drift and correctly re-associating keypoints after temporary self-occlusions characteristic of complex human actions.

**Video Instance Parsing (VIP).** We further challenge the model with the Video Instance Parsing (VIP) benchmark, which requires dense semantic part propagation. Unlike object-level segmentation, this task demands distinguishing between adjacent intra-object parts such as arms, legs, and hair. For this denser task, we increase the retrieval neighborhood to  $k = 10$ . RVM successfully propagates these fine-grained semantic labels, resulting in temporally coherent part segmentations that respect the underlying human geometry better than frame-independent baselines.

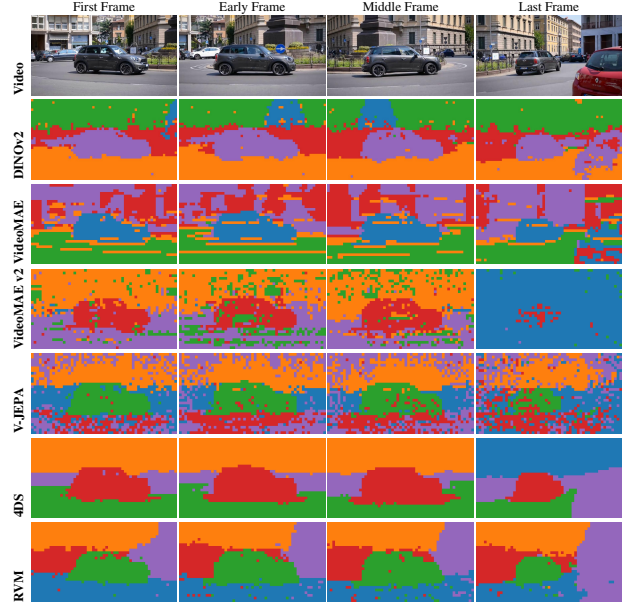


Figure 8. **Temporal Stability in Feature Space.** Using K-Means clustering ( $k = 5$ ) on the car-roundabout sequence, we observe that RVM (Ours) maintains stable cluster assignments for the moving vehicle and the background throughout the clip. In contrast, VideoMAE v2 and 4DS exhibits significant temporal discontinuity ("flickering"), failing to track the object or background consistently over time.

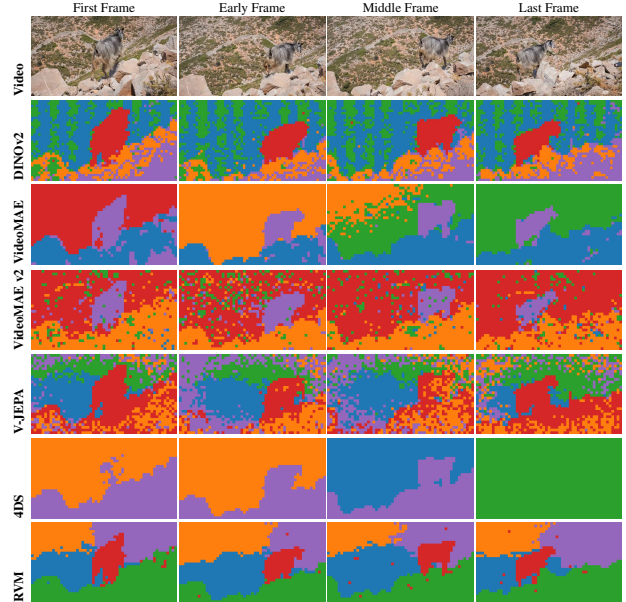


Figure 9. **Robust Foreground-Background Segmentation.** In the goat sequence, RVM effectively disentangles the moving animal from the complex environment. While 4DS suffers from background confusion, merging the object with the scene, RVM produces clean, spatially coherent segments that adhere strictly to object boundaries. DINOv2 segments the object well but fails significantly on the background.

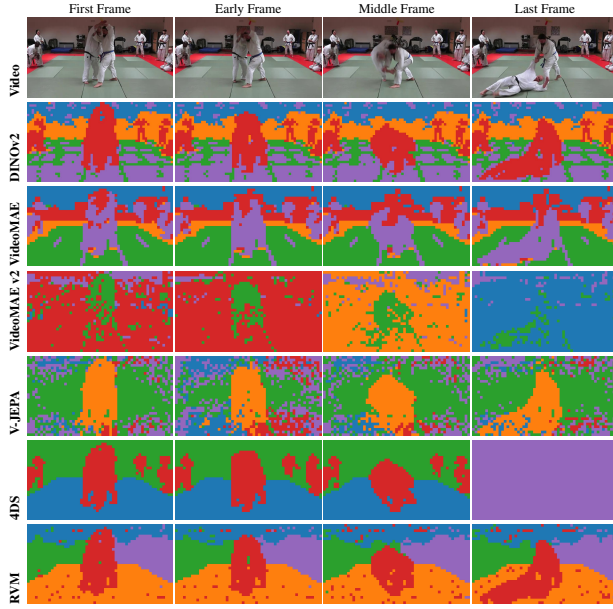


Figure 10. **Motion-Aware Instance Separation.** Visualizing clusters for the judo sequence. **RVM** preserves the structural integrity of semantic parts while separating moving instances from static ones (foreground vs. background human). Notably, it filters out the static background human that **DINOv2** fails to distinguish.

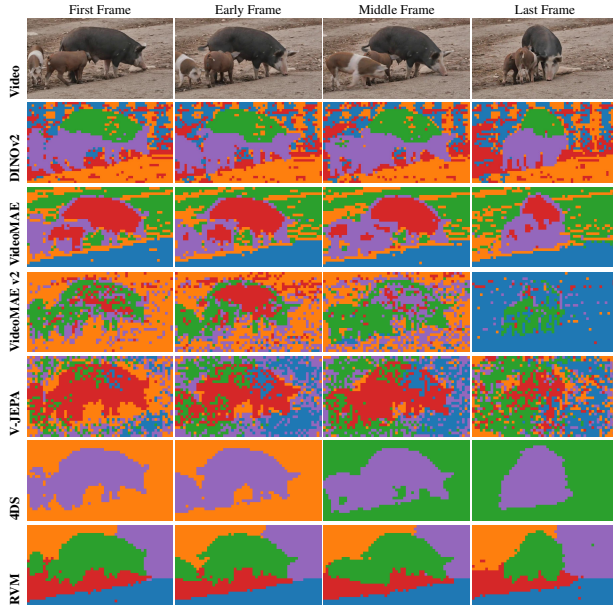


Figure 11. **Long-Term Consistency under Deformation.** Visual results on the pigs sequence demonstrate **RVM**'s ability to maintain consistency over time. While **V-JEPA** exhibits cluster fragmentation, **RVM** leverages recurrent cues to effectively preserve the identity of semantic parts during non-rigid motion.

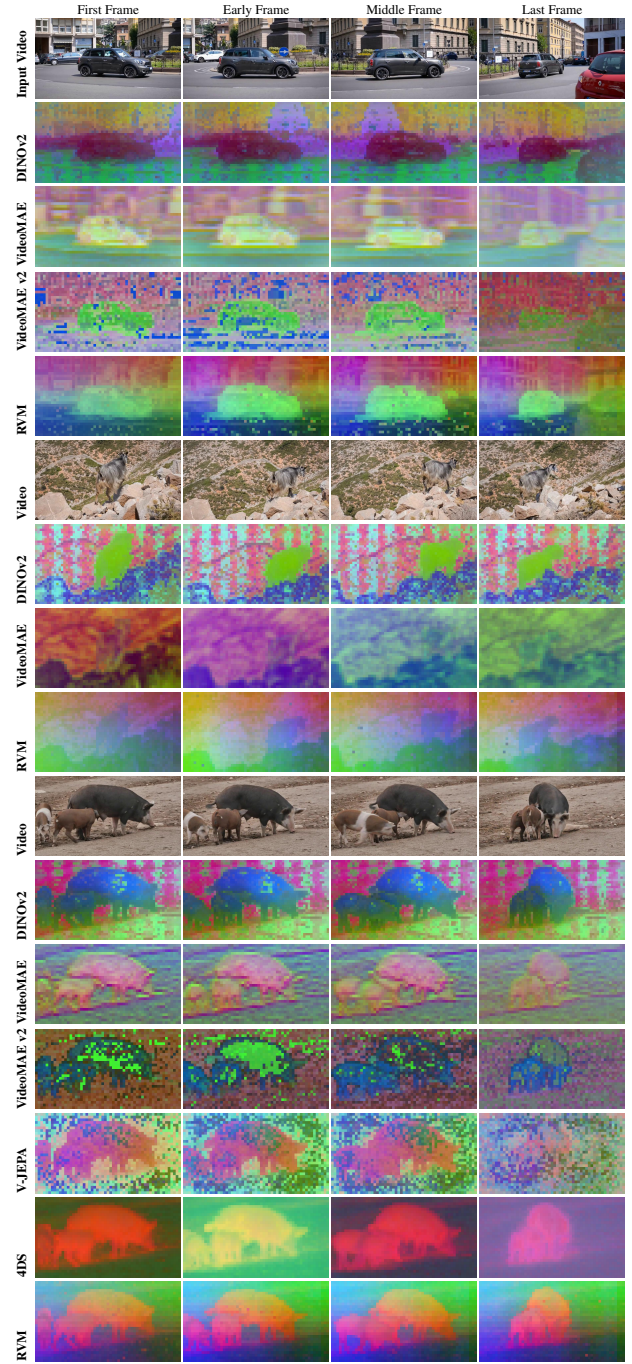


Figure 12. **Intrinsic Dimensionality and Smoothness.** We project the top-3 principal components of the frozen features to RGB space. **RVM** exhibits smooth color gradients that naturally follow the object's geometry, indicating a representation that is both spatially coherent and semantically meaningful. Conversely, features from other models often appear fragmented, lacking clear separation between the foreground and background.



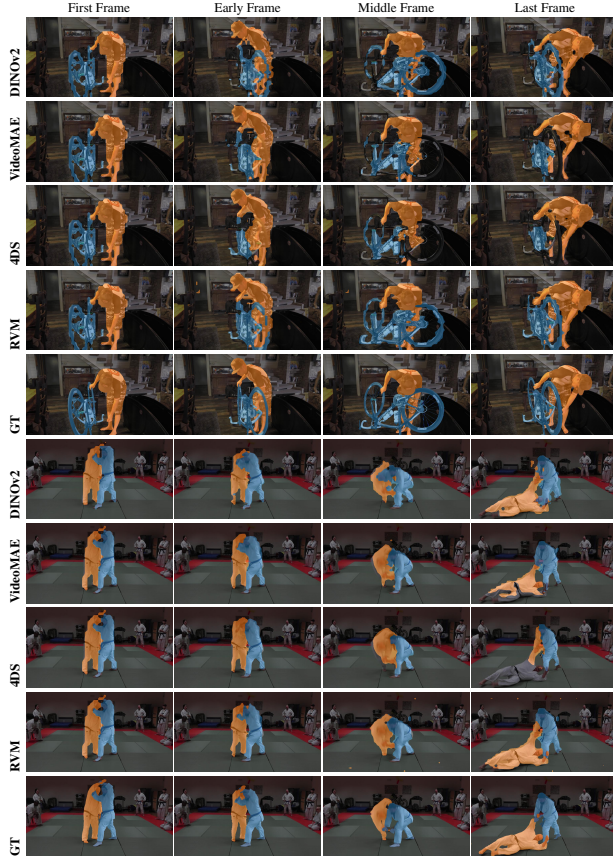


Figure 13. **Qualitative evaluation on DAVIS-2017.** We propagate segmentation masks using nearest-neighbor retrieval ( $k = 7$ ) from a context queue of 20 frames. RVM (Ours) maintains accurate object boundaries and temporal consistency compared to baselines like VideoMAE and 4DS, which often exhibit mask degradation or flickering.

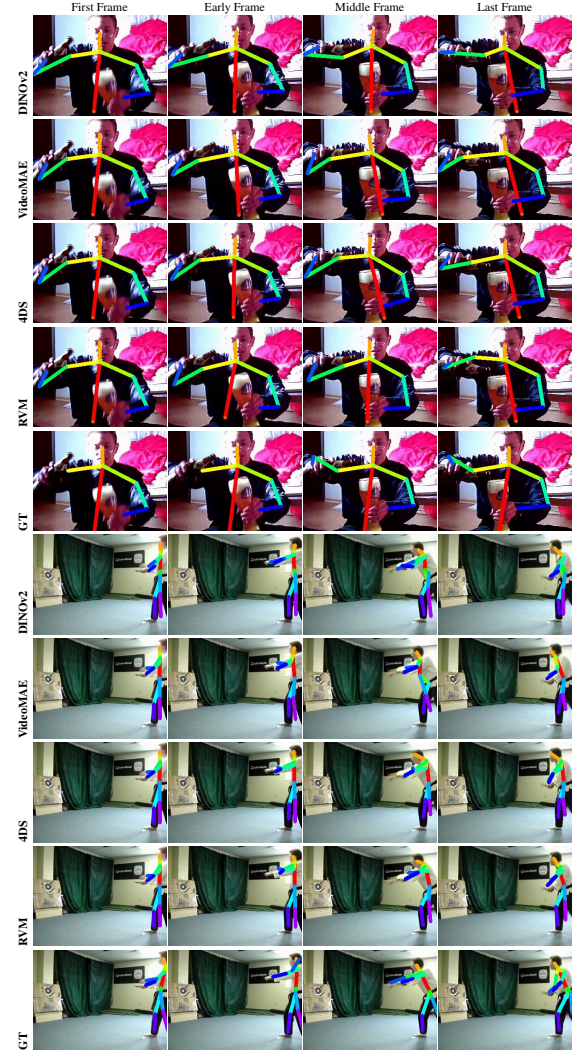


Figure 14. **Keypoint tracking on JHMDB.** The model propagates 15 human joint locations using label propagation ( $k = 7, \tau = 0.7$ ). RVM accurately tracks rapid limb movements and maintains the structural consistency of the pose, distinguishing left/right limbs more effectively than baseline models like VideoMAE and 4DS.



Figure 15. **Semantic part propagation on VIP.** We visualize the propagation of dense part labels (arm, leg, hair, etc.) using  $k = 10$  nearest neighbors. RVM distinguishes fine-grained semantic parts and tracks them consistently across the video clip, whereas other methods often confuse adjacent parts (e.g., arm vs. torso).