TrackMAE: Video Representation Learning via Track Mask and Predict

Supplementary Material

7. Detailed SOTA Comparison

In the main paper, we restricted comparisons to self-supervised masked video models with the same backbone and similar pretraining schedule. We now broaden that comparison to include both self-supervised (MVM variants with pixels/HOG/DINO/CLIP targets) and supervised baselines on K400 and SSv2, reported with the ViT-B backbone and a range of pretraining epochs. We evaluate in the full finetuning setup on K400 and SSv2 datasets to assess in-domain and cross-domain transfer. We also evaluate the data and model scaling behaviors of TrackMAE. The results are shown in Tab. 9.

Kinetics-400 (in-domain pretraining and finetuning). On K400, TrackMAE attains 83.9 Top-1 with 800 epochs, outperforming all listed masked-video baselines trained for equal or longer schedules: VideoMAE [49] (80.0/81.5 at 800/1600), CMAE-V [35] (80.2/80.9), OmniMAE (80.8), MGM (80.8/81.7), MGMAE [23] (81.2/81.8), MME [45] (81.5/81.8), and SIGMA [40] (81.5). TrackMAE also edges SMILE [48] (83.1 at 600 and 83.4 at 1200), indicating a stronger accuracy–compute trade-off at shorter schedules. Compared to supervised architectures, TrackMAE surpasses MViTv2-B [33] (82.9) and Uniformer-B [30] (83.0), despite using the standard ViT-B backbone and a self-supervised objective.

SSv2 (cross-domain: K400→SSv2). When pretrained on K400 and finetuned on SSv2, TrackMAE reaches 72.8, improving over SMILE [48] (72.1 at 600; 72.4 at 1200) and clearly ahead of VideoMAE [49] (68.5), SIGMA [40] (71.1), MGM [14] (71.1), and MGMAE [23] (68.9). It also exceeds the supervised counterparts reported under the same column (e.g., VideoSwin [34] 69.6, MViTv2-B [33] 70.5, Uniformer-B [30] 71.2). These results indicate better domain transfer to motion-centric SSv2, consistent with the hypothesis that explicit motion supervision complements CLIP-space semantics.

SSv2 (in-domain: SSv2→SSv2). With SSv2 pretraining, TrackMAE attains 72.5, matching the best reported MVM result in the table (SMILE [48] at 72.5) and exceeding other MVM baselines like MGMAE [23] (72.0 at 1600), MGM [14] (71.8 at 1600), CMAE-V [35] (70.5 at 1600), VideoMAE [49] (69.6 at 800), and OmniMAE [17] (69.5). This suggests TrackMAE's motion signal remains beneficial even when the pretraining domain already contains substantial temporal variation.

Method scaling. We scaled TrackMAE for both the dataset and model sizes. We pretrained our model on K700, with a ViT-L backbone, using ViT-L CLIP features as spatial targets. We report an improvement of $+1.7\%$ and $+1.5\%$ over VideoMAE with the same backbone, and $+2.9\%$ and $+3.1\%$ over our ViT-B model, showing that our model is able to scale well on larger dataset and model sizes.

In summary, TrackMAE achieves state-of-the-art results among masked-video models under comparable settings and is competitive with, or outperforms, strong supervised models. The pattern supports the central claim that explicit motion supervision paired with CLIP-space reconstruction produces video representations that are both semantically strong and motion-aware, yielding robust transfer in-domain and under-domain shift.

8. Additional Ablations

Impact of masking. Table 10a shows the impact of the masking ratio used during pretraining on the finetuning performance. We observe that a masking ratio of 90%, similar to previous methods [23, 49] works best for the pixel reconstruction. However, we found that for clip reconstruction, a masking ratio of 80% works better, also observed in SMILE [48].

Impact of loss balancing. We analyze the effect of λ with respect to the grid size for CLIP feature reconstruction. The results are shown in Tab. 10b. Without upsampling, the best results are achieved with $\lambda = 1.0$. However, when we upsample from $14 \rightarrow 28$, the best value is obtained with $\lambda = 0.25$. In practice, we favor the upsampling version for efficiency reasons, as already mentioned in the main paper.

Impact of noisy tracks. As mentioned in the main paper, our method shows robustness against noisy target tracks. The results are shown in Tab. 10c. We observe that our method is robust for both spatial and temporal noise, meaning that even under a noisy tracker, our model would still be able to learn meaningful features.

Impact of feature targets. Table 10d highlights how our method behaves when reconstructing DINO features [8] instead of CLIP features. CLIP features work best, but results obtained with DINO features show seamless integration of other type of features.

Impact of point trackers. Similar to Tab. 10c, we analyze in Tab. 10e the influence of the point tracker used for trajectory generation. We observe that TrackMAE is generalizable to different types of point trackers, as TAPNext [58] shows reasonable performance without any tweaking. This demonstrates the usability of the future point trackers as a plugin in the TrackMAE framework.

Table 9. **Detailed SOTA comparison of masked video modeling methods on Something-Something V2 and Kinetics-400 for full finetuning action recognition.** Our TrackMAE outperforms many supervised approaches and achieves the best performance masked video modeling methods with similar pretraining setups. † means we pretrained on K700.

				SSv2 Pretraining	K400 Pretraining	
Method	**Backbone**	**Targets**	**Epochs**	**SSv2 Top-1**	**SSv2 Top-1**	**K400 Top-1**
Supervised						
Mformer [37]	Mformer-B	-	-	-	66.7	79.7
VideoSwin [34]	Swin-B	-	-	-	69.6	80.6
TimeSformer [4]	ViT-B	-	-	-	59.5	80.7
MViTv1 [15]	MViTv1-B	-	-	-	67.7	80.2
MViTv2 [33]	MViTv2-B	-	-	-	70.5	82.9
Uniformer-B [30]	Uformer-B	-	-	-	71.2	83.0
Self-supervised						
VideoMAE [49]	ViT-B	Pixel	800	69.6	68.5	80.0
VideoMAE [49]	ViT-B	Pixel	1600	69.6	-	81.5
CMAE-V [35]	ViT-B	Pixel	800	69.7	-	80.2
CMAE-V [35]	ViT-B	Pixel	1600	70.5	-	80.9
OmniMAE [17]	ViT-B	Pixel	800	69.5	69.0	80.8
MGM [14]	ViT-B	Pixel	800	70.6	71.1	80.8
MGM [14]	ViT-B	Pixel	1600	71.8	-	81.7
MGMAE [23]	ViT-B	Pixel	800	71.0	68.9	81.2
MGMAE [23]	ViT-B	Pixel	1600	72.0	-	81.8
MME [45]	ViT-B	HOG	800	70.0	70.5	81.5
MME [45]	ViT-B	HOG	1600	-	-	81.8
SIGMA [40]	ViT-B	DINO	800	71.2	71.1	81.5
SMILE [48]	ViT-B	CLIP	600	**72.5**	72.1	83.1
SMILE [48]	ViT-B	CLIP	1200	-	72.4	83.4
TrackMAE (Ours)	ViT-B	CLIP	800	**72.5**	**72.8**	**83.9**
VideoMAE [49]	ViT-L	Pixel	1600	-	74.0	85.2
TrackMAE † (ours)	ViT-L	CLIP	800	-	**75.7**	**86.7**

Impact of separate decoding. We study in Tab. 10f the impact of training with a joint or separate decoders to reconstruct both spatial and trajectory targets. As shown in the table, separate decoders work best, which may indicate that involving the same decoder for both tasks may lead to information leakage, degrading the signal.

9. Convergence Results

We analyze the convergence behavior of our approach compared to the baseline. We evaluate our CLIP + Trajectory and CLIP-only reconstruction at different pretraining epochs on both K400 and SSv2 finetuning tasks. Figure 4 shows that our model trained with both the spatial and trajectory targets consistently outperforms the model trained with the spatial targets only. We observe that with fewer epochs, our performance is much better than the baseline; however, the gap is narrowed with more pretraining epochs.

10. Discussion on Motion Masking

In this section, we expand the discussion on our motion-aware strategy. In particular, we expand the discussion on different sampling distributions in Sec. 10.1, and the different sampling strategies in Sec. 10.2 based on motion trajectories.

Table 10. **Additional ablations on our proposed TrackMAE.** The default setting uses pixel reconstruction and motion prediction with a grid size of 14, masking ratio of 90%, $\lambda = 1$, and two separate decoders.

Ratio	$K400_s$	$SSv2_s$
95%	48.1	54.4
90%	**49.4**	**56.2**
80%	49.2	56.0

(a) **Masking ratio.** Masking 90% of the tokens, as in previous methods, works best.

λ	GridSize	$K400_s$	$SSv2_s$
0.25	28	54.4	59.6
0.5	28	55.1	60.1
1.0	28	55.8	61.1
0.25	$14 \rightarrow 28$	**55.9**	**61.1**
0.50	$14 \rightarrow 28$	55.0	60.4
1.0	$14 \rightarrow 28$	54.3	59.7

(b) **Sensitivity of λ.** For CLIP reconstruction and upsampled trajectory prediction, $\lambda = 0.25$ achieves the best performance.

Noise	$K400_s$	$SSv2_s$
None	**49.4**	**56.2**
Spatial	48.9	55.7
Temporal	48.8	55.8

(c) **Impact of noisy tracks.** Adding spatial or temporal noise to the tracker trajectories does not affect our method much.

Method	$K400_s$	$SSv2_s$
TrackMAE-DINO	55.2	60.6
TrackMAE-CLIP	**55.8**	**61.1**

(d) **Target type comparison**. Reconstructing DINO or CLIP features works well for spatial reconstruction signal.

Method	$K400_s$	$SSv2_s$
TAPNext	55.3	60.7
CoTracker3	**55.8**	**61.1**

(e) **Tracker comparison**. Leveraging different off-the-shelf point tracker leads to overall close performance.

Decoder	$K400_s$	$SSv2_s$
Joint	48.7	55.5
Separate	**49.4**	**56.2**

(f) ***Joint* or *Separate* decoders**. Using separate decoder for motion prediction outperforms using the joint decoder.

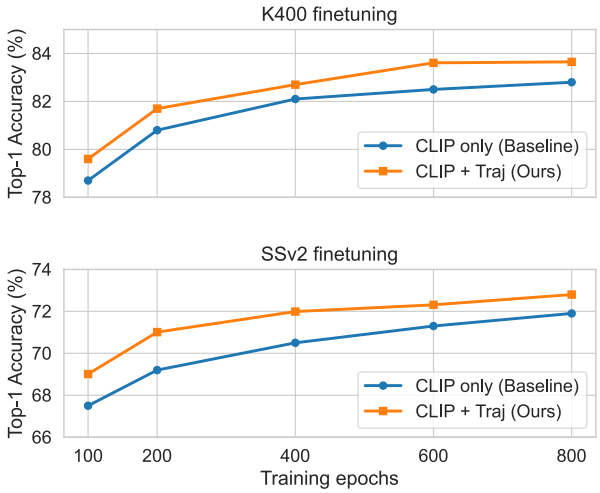

Figure 4. **Training evolution.** We report the Top-1 Accuracy for K400 and SSv2 finetuning at different pretraining epochs. Our model trained with both CLIP and trajectory reconstructions consistently outperforms the CLIP reconstruction only.

Figure 5. **Sampling strategies.** We show how we can use the motion information to create different sampling distribution.

10.1. Sampling Distribution in Masking

As mentioned in the main paper, we use the motion trajectories to create a sampling distribution. We explore two different ways to do this: (1) we accumulate the displacement made by each point of the grid through time with respect to the first frame, (2) we accumulate the displacement made by each point of the grid through time in the next consecutive frame. In other words, the first strategy gives the average motion information over time of *how much* a given point is moving, without taking into account the trajectory. The second strategy gives the average motion information over time of *how much* and *where* a given point is moving. Those 2 sampling strategies are depicted in Fig. 5.

Both sampling strategies have their own merits, *i.e.* the first strategy will put some emphasis towards tokens that are likely to move, without guarantee that they are seen in the following frames. Our intuition is that it forces the model towards learning some motion dynamics of the different moving objects. For the second strategy, our intuition follows the same reasoning, but with visible tokens sampled along the trajectory. In practice, we see both version works on par, with slightly better results for the first strategy, which is used for the results in the main paper. It is also worth mentioning that whatever the strategy used, it is always impacted by the input data. We observe that the K400 dataset is prone to have erratic movements, which may lead to poor motion information. In such a case, our sampling distribution tends to behave more like a uniform sampling, thus falling back to the original random tube masking strategy.

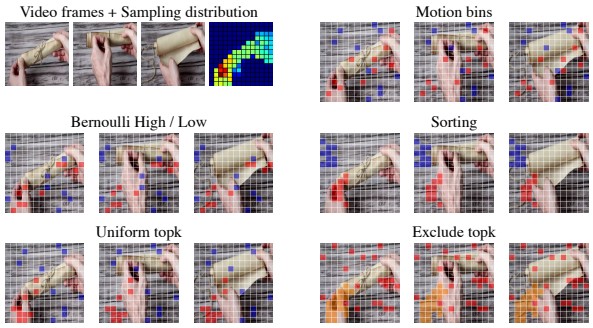

Figure 6. **Masking strategies.** We show how we can use our sampling distribution to create different masking maps. Red squares are high-motion visible tubes, blue squares are low-motion visible tokens. For the "exclude topk" strategy, we also show orange the tokens excluded from being sampled.

10.2. Sampling Strategy in Masking

Besides different sampling strategy, there are also many different ways to sample visible tokens from it. For computational reasons and for its intuitive soundness, we only evaluated the motion bins strategy described in the main paper. However, we describe below some other strategies.

Motion bins. We sample visible tokens from high- and low-motion regions of the sampling distribution using 2 uniform bins. Depending on the value of ρ_{motion}, we control the number of tokens sampled coming from each bins. Our intuition is that this strategy gives, on average, a good balance between high- and low-motion visible and masked tokens. This is the strategy used in the main paper.

Bernoulli High/Low. Instead of uniformly sampling from high- and low-motion regions, we can sample using a Bernoulli distribution. For the high-motion tokens, we directly sample from the sampling distribution, and for the low-motion token, we sample from the "1-motion" distribution, excluding tokens already sampled from the high-motion part. Similarly as for the motion bins strategy, we can use the same parameter to control the number of tokens for each high- and low-motion regions. However, we visually observe that this strategy tends to create blob regions, which would usually hurts training.

Sorting. Based on the previous strategy, we can directly sort high- and low-motion tokens and follow their ordering, instead of sampling from the distribution. This strategy creates bigger blobs, which we believe would hurt even more the training.

Uniform topk. To find a better balance between Bernoulli and sorting, we can uniformly sample from the Top-k most moving tokens. However, in order to sample from a smaller set than in the motion bins strategy, we need to specify how many samples are used in the Top-k operation, adding a new hyperparameter to tune.

Exclude topk. In contrast to previous strategies, which use the motion to guide the sampling of visible tokens, we can use the motion information to decide where *we should not* sample visible tokens. From the Top-k most moving tokens, we can chose to explicitly remove them from the sampling distribution, as learning to reconstruct those tokens may be interesting. Then, we can use any strategy to sample the visible tokens. All sampling strategies are shown in Fig. 6

We leave the exploration of such masking strategies, their impact on different pretraining data and downstream tasks for future work.

11. Experimental Details

11.1. Datasets

Kinetics-400 [27] (K400) is a large-scale YouTube-sourced corpus with 400 human action categories and over 306k short clips. It remains a canonical benchmark for learning generalizable video representations.

Something-Something V2 [19] (SSv2) emphasizes object-centric, first-person interactions that differ markedly from K400's Internet footage. It comprises 168,913 training and 24,777 test samples spanning 174 categories, stressing temporal reasoning and commonsense dynamics.

UCF-101 [44] (UCF) collects 9,537 training and 3,783 test clips from YouTube across 101 action classes. Although coarser in granularity and overlapping with K400 categories, it is widely used for transfer evaluation in self-supervised video learning.

HMDB-51 [28] (HMDB) contains 6,766 clips from various sources (films, archives, web videos) covering 51 classes (at least 100 videos per class). Its heterogeneity challenges models to cope with diverse cinematic and real-world content.

FineGYM [42] (GYM) targets fine-grained action understanding in gymnastics. We use the Gym-99 subset (99 classes) with 20,484 training and 8,521 test samples, focusing on subtle motion differences within highly structured routines.

SEVERE Benchmark [46] SEVERE aggregates eight evaluation settings across SSv2, UCF, FineGYM, and Charades to stress sample efficiency, granularity, and task shift. Table 15 details each subset and metric.

A summary of the different datasets is presented in Tab. 14.

11.2. Training and Evaluation Details

Pretraining Details. We pretrain on K400 [27] and SSv2 [19]. Following VideoMAE [49], we sample clips of 16 frames at 224×224 with a temporal stride of 2 on SSv2 and 4 on K400. We compute space–time tube tokens

Table 11. **Pretraining configuration.**

Shared	
Optimizer	AdamW
Base learning rate	1.5×10^{-4}
Weight decay	0.05
Momentum (Betas)	β_1=0.9, β_2=0.95
Batch size	512
LR schedule	cosine decay
Warmup epochs	40
Augmentation	MultiScaleCrop(1, 0.875)

Dataset-specific	**Epochs**	**FlipAug.**
SSv2	800	no
K400	800	yes

Table 12. **Linear probing configuration.**

config	K400	HMDB	SSv2	GYM
optimizer	AdamW			
base learning rate	1×10^{-3}			
weight decay	0.05			
optimizer momentum	$\beta_1, \beta_2 = 0.9, 0.999$			
layer-wise lr decay	0.75			
batch size	128			
learning rate schedule	cosine decay			
training epochs	30	100	50	100
flip augmentation	*yes*	*yes*	*no*	*yes*

Table 13. **Full finetuning configuration.**

Parameter	**Value**		
Optimizer	AdamW		
Base learning rate	1.0×10^{-3}		
Weight decay	0.05		
Momentum (Betas)	$\beta_1 = 0.9,\ \beta_2 = 0.999$		
Layer-wise LR decay	0.75		
LR schedule	cosine decay		
Warmup epochs	5		
RandAug	(9, 0.5)		
Label smoothing	0.1		
Mixup	0.8		
CutMix	1.0		
Drop path	0.1		
Dataset-specific	**Batchsize**	**Epochs**	**FlipAug.**
SSv2	32	40	no
K400	16	100	yes
SEVERE	16	100	yes

via a 3D convolution, treating each 2×16×16 cube as a token. For each sampled 16-frame clip, we temporally downsample it with a stride of 2 to extract the trajectories from CoTracker3 [26], matching the total number of trajectory tokens with space-time cubes. In practice, we use the normalized temporal differences of the extracted motion trajectories as the target, rather than absolute values.

Table 14. **Datasets details.** Splits of datasets used in full finetuning and linear probing.

Dataset	**Abbrev.**	**#Classes**	**#Train**	**#Test**
Kinetics-400	K400	400	240K	19K
UCF-101	UCF	101	9.5K	3.8K
HMDB-51	HMDB	51	4.8K	2K
Something-Something V2	SSv2	174	169K	24.8K
FineGYM	GYM	99	20.5K	8.5K

All hyperparameters are shown in Tab. 11, following [48, 49]. All pretraining runs use 16×NVIDIA A100 GPUs. For downstream tasks, we discard the decoders and use only the pretrained encoder with a task-specific head (*e.g.*, a linear classifier for action recognition).

Linear probing details. We strictly follow the settings in [48] and train the linear head on top of the frozen backbone with the target dataset, as shown in Tab. 12. Experiments are run on 4×V100 GPUs.

Full finetuning details, We strictly follow the settings in [48] and train the backbone + head with the taget dataset, as shown in Tab. 13. Experiments are run on 4×V100 GPUs.

SEVERE benchmark details. Following [40, 48], we evaluate with the official SEVERE codebase [46], strictly reusing the provided training and evaluation configurations to ensure a fair comparison, also shown in Tab. 13. Experiments are run on 4×V100 GPUs.

Table 15. **SEVERE benchmark.** Subsets, protocols, and metrics following [46].

Dataset	Experiment	Setup Group	Task	#Classes	Finetune	Test	Metric
FineGym [42]	Gym99	Full	Action Class.	99	20,484	8,521	Top-1 Acc.
UCF 101 [44]	UCF (10^3)	Sample Efficiency	Action Class.	101	1,000	3,783	Top-1 Acc.
FineGym [42]	Gym (10^3)	Sample Efficiency	Action Class.	99	1,000	8,521	Top-1 Acc.
FineGym [42]	FX-S1	Action Granularity	Action Class.	11	1,882	777	Mean-per-class
FineGym [42]	UB-S1	Action Granularity	Action Class.	15	3,511	1,471	Mean-per-class
UCFRep [57]	UCF-RC	Task Shift	Repetition Counting	–	421	105	Mean Error
Charades [43]	Charades	Task Shift	Multi-label Class.	157	7,985	1,863	mAP