InternVideo-Next: Toward World Understanding Video Models

Chenting Wang[1,2] Yuhan Zhu[2,5] Yicheng Xu[2]
Jiange Yang[2,5] Ziang Yan[2] Yali Wang[2,4] Yi Wang[2,3] Limin Wang[1,2,5,♠]

[1]Shanghai Jiao Tong University [2]Shanghai AI Laboratory [3]Shanghai Innovation Institute
[4]Shenzhen Institutes of Advanced Technology, China [5]State Key Laboratory for Novel Software Technology, Nanjing University

Method	Memory(G)	K400
SigLip Align Only.	22	70.7
Align + Linear Recon.	41	69.8
Align + Diffusion Recon.	48	75.8

Table 1. **GPU memory utilization comparison.** Models are trained with the Stage 1 settings using 16 frames and a mask ratio of 80%. We use a batch size of 16 and report the memory consumption on a single A100 GPU.

A. Training GPU utilization

Table 1 shows memory consumption using different training strategies in Stage 1. Utilizing a patch diffusion decoder increases the memory consumption to an acceptable level, considering the performance boost it brings. We further explain the process of diffusion loss in Algorithm 1.

B. More implementation details

In this section, we introduce the model architectures, training hyperparameters, and test settings in our experiments.

B.1. Model architecture and training details

We use a patch of size 14 for our InternVideo-Next models. We report the detailed architecture of a Base-scaled model in Table 2, and the pre-training details in Tables 3, 4.

B.2. Action Recognition

We show the detailed hyperparameters for action recognition probing in Table 5. We use an attention pooling head that averages the input tokens into a single token. And the single token is utilized as the common [CLS] token for classification, as shown in Figure 1 (a).

B.3. Depth Estimation

Experiment Setup. To assess the spatiotemporal, low-level, and 3D-geometry related capabilities of InternVideo-Next, we evaluate it on two widely used monocular video depth-estimation benchmarks: ScanNet and KITTI. ScanNet is a large-scale RGB-D video dataset of indoor scenes, containing over 2.5 million views across more than 1500 scans with rich annotations. KITTI is an outdoor dataset collected using an autonomous-driving platform, where stereo videos are recorded by high-resolution color cameras and depth ground truth is captured via a Velodyne LiDAR sensor. Throughout experiments, we adopt a head-probing setup in which the backbone network remains frozen.

Stage	ViT-B	Output Size
Data	sparse sampling	$3\times8\times224\times224$
Patch Embedding	$1\times14\times14$, 768 stride $1\times14\times14$	$768\times8\times256$
Position Embedding	sine-cosine 768×2048	768×2048
Mask	semantic mask *mask ratio* $= \rho$	$768\times2048\cdot(1\text{-}\rho)$
Encoder	[MHSA(768) MLP(3072)] $\times12$	$768\times2048\cdot(1\text{-}\rho)$
Projection	[LN(768) MLP(1408)] $\times K$	$K\times1408\times2048\cdot(1\text{-}\rho)$

Table 2. **Architecture of video encoder.** We take ViT-B with 8-frame input as an example. "MHSA", "MLP" and "LN" refer to spatiotemporal multi-head self-attention, multi-layer perceptron and layer normalization. K means the layer number for unmasked token alignment. We mark the channel number, frame number, spatial size and token number by different colors.

config	SthSth V2	Others
optimizer	AdamW [6]	
optimizer momentum	β_1, β_2=0.9, 0.98	
weight decay	0.05	
learning rate schedule	cosine decay [7]	
learning rate	1e-3	
batch size	2048	
warmup epochs [5]	5	
total epochs	50	
input frame	16	
spatial resolution	224	
flip augmentation	*no*	*yes*
augmentation	MultiScaleCrop [0.66, 0.75, 0.875, 1]	

Table 3. **InternVideo-Next Stage 1 pre-training settings.**

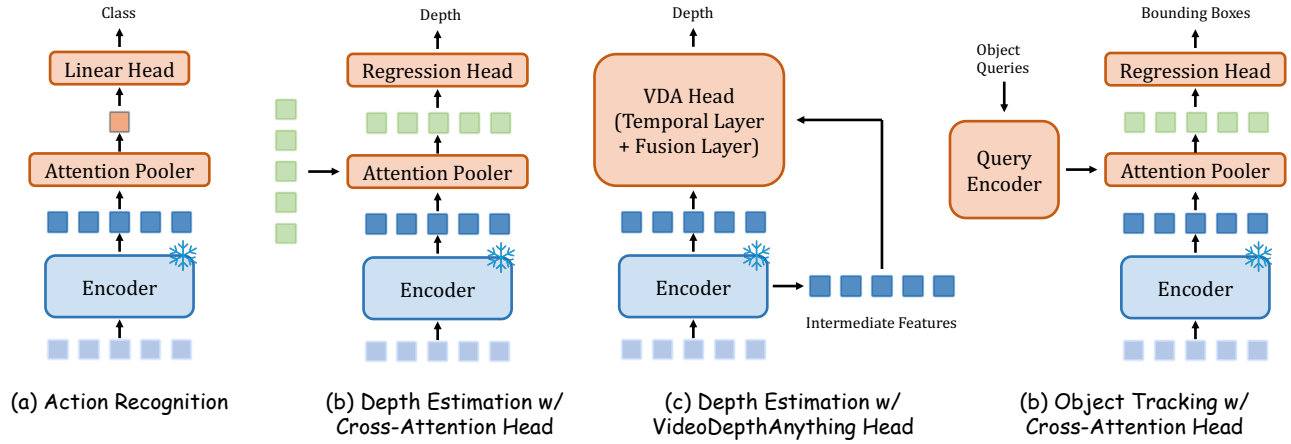

Figure 1. **Explanations of different probing settings in our paper.**

```
class DiffusionLoss(nn.Module)
   def __init__(depth = 6, width = 1536):
      # SimpleMLP takes in x_t, timestep, and condition, and outputs predicted noise.
      self.net = SimpleMLP(depth, width)

      # GaussianDiffusion offers forward and backward functions q_sample and p_sample.
      self.diffusion = GaussianDiffusion()

   # Given condition z and ground truth token x, compute loss
   def loss(self, z, x):
      # sample random noise and timestep
      noise = torch.randn(x.shape)
      timestep = torch.randint(0, self.diffusion.num_timesteps, x.size(0))

      # sample x_t from x
      x_t = self.diffusion.q_sample(x, timestep, noise)

      # predict noise from x_t
      noise_pred = self.net(x_t, timestep, z)

      # L2 loss
      loss = ((noise_pred - noise) ** 2).mean()

      return loss
```

Algorithm 1 Pseudo-code illustrating the concept of Diffusion Loss. Here, the conditioning vector z is the output from the Predictor. The gradient is backpropagated to z. The GT x is generated by patching the video input with a pixel shuffle function and a patch size of 14. And then we input the masked part of x and the predicted part z into the model.

Training Details. To provide a comprehensive evaluation, we employ two representative prediction heads: (1) a *Simple* probing head, following Scaling4d[2], implemented as a lightweight cross-attention pooling layer with an embedding dimension of 1536 (see Figure 1(b)); (2) a *Temporally Aware* head [3] based on VideoDepthAnything, consisting of four stacked temporal modules specifically designed to adapt image models such as DINO to video depth-estimation tasks (see Figure 1(c)). As it contains spatial modules tailored for depth prediction, we also apply it to video models. We use a learning rate of 3×10^{-3} with batch sizes of 128 (KITTI) and 256 (ScanNet). Models are trained for 30 epochs, which we find sufficient for convergence. A sliding-window strategy is applied to segment input videos, using window sizes of 32 (stride 10) for KITTI and 16 (stride 5) for ScanNet. Following standard practice, the valid depth range is set to $[0.1, 80.0]$ for KITTI and $[0.001, 10.0]$ for ScanNet, with values outside these intervals masked or clamped.

Competitive Methods. We compare InternVideo-Next with state-of-the-art image and video representation models, re-implementing all baselines under a unified training and evaluation protocol for fairness. For image-based models, we include SigLip2 and DINOv3, known for strong se-

config	SthSth V2	Others
optimizer	AdamW [6]	
optimizer momentum	β_1, β_2=0.9, 0.98	
weight decay	0.05	
learning rate schedule	cosine decay [7]	
learning rate	1e-4	
batch size	2048(Base) 1024(Large)	
warmup epochs [5]	0	
total epochs	100	
input frame	32	
spatial resolution	224	
flip augmentation	*no*	*yes*
augmentation	MultiScaleCrop [0.66, 0.75, 0.875, 1]	

Table 4. **InternVideo-Next Stage 2 pre-training settings.**

config	SthSth v2	Kinetics	Coin
optimizer	AdamW [6]		
optimizer momentum	β_1, β_2=0.9, 0.999		
learning rate schedule	cosine decay [7]		
learning rate	1e-3	3e-4	2e-4
batch size	1024		
repeated augmentation	2		
warmup epochs [5]	5		
total epochs	30	20	20
layer-wise lr decay [1]	0.75 (B), 0.85 (L)		
flip augmentation	*no*	*yes*	*yes*
label smoothing [8]	0.1		
cutmix [9]	1.0		
augmentation	RandAug(9, 0.5) [4]		

Table 5. **Action recognition fine-tuning settings.**

mantic alignment and high-quality visual representations. For video-based models, we evaluate VideoMAEv2, InternVideo2, and V-JEPA, representing the current state of the art in video representation learning.

Evaluation Metrics. Following standard monocular depth-estimation protocols, we report two metrics that jointly assess geometric accuracy.

1. Absolute Relative Error (AbsRel):

$$\text{AbsRel} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{|D_i - \hat{D}_i|}{D_i}\right),$$

where D_i and $\hat{D}_i$ denote the ground-truth and predicted depths, respectively. This metric reflects the average relative deviation of predictions from the ground truth.

2. δ_1: The proportion of predictions within a factor of 1.25 of the ground truth, providing a measure of prediction robustness and can be viewed as the accuracy rate.

B.4. Object Tracking

Setup. To evaluate the model's capability to capture object-level motion, we conduct experiments on the Waymo Open dataset, following the protocol of Scaling4D [2]. The videos include 2D and 3D bounding-box annotations. We use only the RGB frames as model input and the 2D bounding boxes for computing losses and evaluation metrics. Because the original Scaling4D benchmark construction code is not publicly available, we reconstruct the training and test sets based on descriptions provided in their paper. Given the target objects' bounding boxes in the first frame, the models are required to output their bounding boxes in the subsequent frames.

Preprocessing. Raw 1280×1920 RGB videos are spatially downsampled to 224×336 and then centrally cropped to 224×224. All bounding boxes are remapped to the cropped coordinate space, and boxes occupying less than 0.5% of the cropped frame area are filtered out. Temporally, the original 20-second sequences recorded at 10 fps are further downsampled to 5 fps.

Dataset Construction. Training samples are generated via a sliding window of 16 frames (length $=$ 16, stride $=$ 1) applied to each downsampled video. To avoid excessive redundancy, the starting indices for extracted windows must be at least three frames apart. We impose several object-consistency constraints: each window must contain 1–25 objects in the first frame; each first-frame object must appear in at least 70% of frames within the window; all first-frame objects must co-occur in at least 10 frames; and each object may have at most a continuous 5-frame missing gap. No further area-based filtering beyond the initial 0.5% threshold is applied.

Training and Evaluation. During training, the backbone is frozen and only the probing head is fine-tuned. The head encodes the initial bounding boxes—specifying the objects to track—using a query encoder. The resulting query tokens are fed into a one-layer cross-attention module to obtain pooled tokens, which are then decoded linearly to regress bounding boxes for all frames except the first. The training objective is a weighted sum of Smooth L1 loss (1.0) and GIoU loss (0.5). For evaluation, the Intersection-over-Union (IoU) is averaged across all objects and frames.

References

[1] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers. In *ICLR*, 2021. 3

[2] João Carreira, Dilara Gokay, Michael King, Chuhan Zhang, Ignacio Rocco, Aravindh Mahendran, Thomas Albert Keck, Joseph Heyward, Skanda Koppula, Etienne Pot, Goker Erdogan, Yana Hasson, Yi Yang, Klaus Greff, Guillaume Le Moing, Sjoerd van Steenkiste, Daniel Zoran, Drew A. Hudson, Pedro Vélez, Luisa Polanía, Luke Friedman, Chris Duvarney, Ross Goroshin, Kelsey Allen, Jacob Walker, Rishabh Kabra,

Eric Aboussouan, Jennifer Sun, Thomas Kipf, Carl Doersch, Viorica Pătrăucean, Dima Damen, Pauline Luc, Mehdi S. M. Sajjadi, and Andrew Zisserman. Scaling 4d representations. *arXiv*, 2025. 2, 3

[3] Sili Chen, Hengkai Guo, Shengnan Zhu, Feihu Zhang, Zilong Huang, Jiashi Feng, and Bingyi Kang. Video depth anything: Consistent depth estimation for super-long videos. In *CVPR*, 2025. 2

[4] Ekin Dogus Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020. 3

[5] Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *ArXiv*, abs/1706.02677, 2017. 1, 3

[6] I. Loshchilov and F. Hutter. Fixing weight decay regularization in adam. *ArXiv*, abs/1711.05101, 2017. 1, 3

[7] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *ICLR*, 2017. 1, 3

[8] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 3

[9] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Young Joon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, 2019. 3