DiffuView: Multi-View Diffusion Pretraining for 3D-Aware Robotic Manipulation

Supplementary Material

A. Pretraining and Feature Extractor Details

As illustrated in Fig. 8, our multi-view diffusion model employs both 2D spatial attention and 3D full attention. The 2D spatial attention operates independently on each frame and can be written as:

$$\mathbb{R}^{b\times f\times h\times w\times c} \longrightarrow \mathbb{R}^{(bf)\times(hw)\times c}. \tag{11}$$

In contrast, the 3D full attention performs joint spatio-temporal attention over all frames, corresponding to the mapping:

$$\mathbb{R}^{b\times f\times h\times w\times c} \longrightarrow \mathbb{R}^{b\times(fhw)\times c}. \tag{12}$$

It is worth noting that our model does not perform cross-attention with text features, nor do we introduce any fixed positional encoding to the input frames. Consequently, the number of input images can be arbitrary, and the model can be viewed as a pure feature extractor. Formally, given an image, we further select the feature maps from the upsampling blocks at spatial resolutions 8×8 and 16×16 as the inputs to the query transformer. To construct a multi-scale representation for query transformer processing, we fuse features extracted at different spatial resolutions. Specifically, let$F_8 \in \mathbb{R}^{b\times 8\times 8\times c}$ and $F_{16} \in \mathbb{R}^{b\times 16\times 16\times c}$ denote the feature maps at resolutions 8×8 and 16×16, respectively. We first upsample the low-resolution feature map to match the higher spatial resolution, $\widetilde{F}_8 = \mathrm{Upsample}(F_8) \in \mathbb{R}^{b\times 16\times 16\times c}$, and then concatenate the two representations along the channel dimension,$F_{\mathrm{ms}} = \mathrm{Concat}\big(\widetilde{F}_8,\, F_{16}\big) \in \mathbb{R}^{b\times 16^2\times 2c}$, yielding a unified multi-scale fused feature embedding.

We further adopt a FiLM-style modulation mechanism to incorporate global conditioning language signals. Given a conditioning vector $\mathbf{z}$, the FiLM layer applies a feature-wise affine transformation to the fused features:

$$\mathrm{FiLM}(\mathbf{x}) = \gamma(\mathbf{z}) \odot \mathbf{x} + \beta(\mathbf{z}), \tag{13}$$

where $\gamma(\cdot)$ and $\beta(\cdot)$ denote the scale and shift functions.

The FiLM-conditioned multi-scale features are then provided as input to a query transformer equipped with 32 learnable query tokens, $Q \in \mathbb{R}^{b\times 32\times d}$. Let $X = \mathrm{MLP}(F_{\mathrm{ms}}) \in \mathbb{R}^{b\times 16^2\times d}$ denote the feature tokens. The query transformer computes

$$H = \mathrm{QFormer}(Q, X, L), \tag{14}$$

producing the final output embeddings $H \in \mathbb{R}^{b\times 32\times 768}$. For conditioning, we employ the CLIP end-of-text (EoT) token L as the conditioning signal, enabling global semantic context to modulate the learned feature representations.

B. Action Diffusion Policy Details

Given the encoded context y_k (e.g., the observations at time k and language goal embedding), our goal is to model the conditional distribution over future actions $p(A_k \mid y_k)$, where $A_k = a_{k:k+T}$. To generate the future T step actions, we follow the diffusion-based formulation and first corrupt the ground-truth action $A_k^0 = A_k$ with Gaussian noise to obtain a noised action A_k^τ at noise level τ:

$$p^{0\to\tau}\big(A_k^\tau \mid A_k^0\big) = \mathcal{N}\big(A_k^\tau;\, A_k^0,\, \sigma^2(\tau)\mathbf{I}\big)\,, \tag{15}$$

where $\sigma(\tau)$ denotes the noise schedule and $p^{0\to\tau}(\cdot \mid \cdot)$ is the corresponding Gaussian perturbation kernel. This forward diffusion process can be described as the solution to the following stochastic differential equation (SDE):

$$\mathrm{d}A = f(A,\tau)\,\mathrm{d}\tau + g(\tau)\,\mathrm{d}\mathbf{w}, \tag{16}$$

where $\mathbf{w}$ is a standard Wiener process, f is the drift coefficient, and g is the diffusion coefficient. For the simple Gaussian perturbation in Eq. (15), the effect of the diffusion is equivalent to choosing

$$f(A,\tau) = 0, \qquad g(\tau) = \sqrt{2\,\dot\sigma(\tau)\,\sigma(\tau)}, \tag{17}$$

where $\dot\sigma(\tau)$ denotes the derivative of the noise schedule with respect to τ. Starting from a highly noised action sample at large τ, we can then learn a conditional denoising model to reverse this diffusion process and generate clean future actions A_k conditioned on y_k. Following the recommended EDM configuration for modeling moderately scaled clean data, we let $x_t = A_k^t$ denote the noised action corrupted at noise-adding step t. The denoiser D_θ is therefore written as

$$D_\theta(x_t,\sigma_t) = c_{\text{skip}}(\sigma_t)\,x_t + c_{\text{out}}(\sigma_t)\,F_\theta\big(c_{\text{in}}(\sigma_t)\,x_t,\; c_{\text{noise}}(\sigma_t)\big)\,, \tag{18}$$

where F_θ is the neural network operating on normalized inputs and an explicit noise-level embedding. Such preconditioning has been shown to significantly improve numerical stability and reduce the dynamic range requirements of the network output. The detailed design is as follows:

$$c_{\text{in}} = \frac{1}{\sqrt{\sigma_t^2 + \sigma_{\text{data}}^2}}, \tag{19}$$

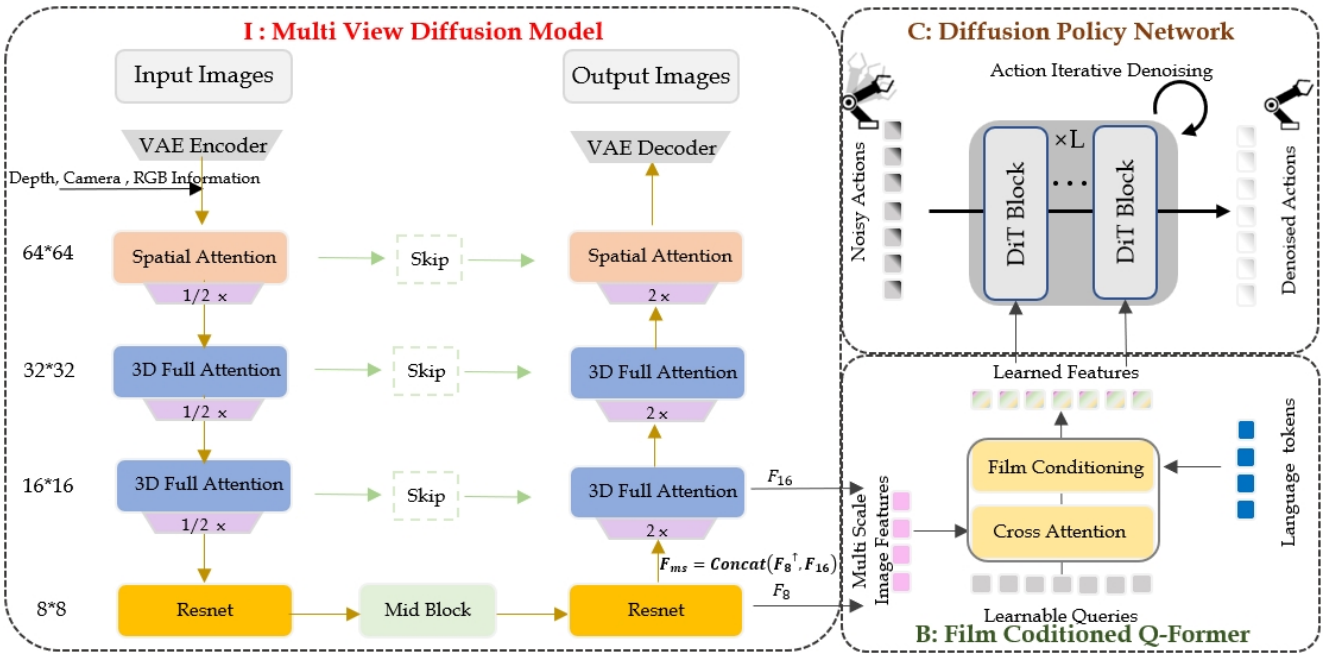

Figure 8. The details of the multi-view diffusion model.

$$c_{\text{out}} = \frac{\sigma_t\,\sigma_{\text{data}}}{\sqrt{\sigma_t^2 + \sigma_{\text{data}}^2}}, \tag{20}$$

$$c_{\text{skip}} = \frac{\sigma_{\text{data}}^2}{\sigma_{\text{data}}^2 + \sigma_t^2}, \tag{21}$$

where σ_t denotes the noise magnitude. We set $\sigma_{\text{data}} = 0.5$, $\sigma_{\min} = 0.001$, and $\sigma_{\max} = 80$. The variable σ_t is sampled according to the truncated log-logistic distribution:

$$a = \text{sigmoid}\left(\frac{\ln(\sigma_{\min}/\sigma_{\text{data}})}{\text{scale}}\right), \tag{22}$$

$$b = \text{sigmoid}\left(\frac{\ln(\sigma_{\max}/\sigma_{\text{data}})}{\text{scale}}\right), \tag{23}$$

$$u \sim \mathcal{U}(a, b), \tag{24}$$

$$\sigma_t = \exp(\text{Logit}(u) \cdot \text{scale} + \ln(\sigma_{\text{data}})), \tag{25}$$

where scale $= 0.5$. During inference, we uniformly select 10 timesteps in the interval $[\ln(\sigma_{\min}), \ln(\sigma_{\max})]$ and apply a DDIM scheduler to traverse this noise schedule. Following the EDM framework, we minimize a noise-level-weighted mean-squared error:

$$\mathcal{L}(\theta) = \mathbb{E}_{x_0,\varepsilon,\sigma_t}\left[w(\sigma_t)\left\|D_\theta(x_t, \sigma_t) - x_0\right\|_2^2\right], \tag{26}$$

where $w(\sigma_t)$ is a scalar weighting function that balances the contributions from different noise levels. In our implementation we adopt the EDM loss weighting

$$w(\sigma_t) = \frac{\sigma_t^2 + \sigma_{\text{data}}^2}{(\sigma_t\,\sigma_{\text{data}})^2}, \tag{27}$$

which emphasizes low-noise regimes while still providing sufficient training signal at higher noise levels, and has been empirically shown to improve convergence and sample quality in EDM-style models.

To further specialize the denoiser across different noise levels, we replace the single EDM denoiser network with a mixture-of-experts (MoE) architecture whose routing mechanism is explicitly conditioned on the current noise magnitude σ_t. Let $\{F^{(i)}\}_{i=1}^N$ denote N denoising experts and let $\eta(\sigma_t) \in \mathbb{R}^D$ be a learnable noise embedding. The router produces a gating vector over experts as

$$\mathbf{r}(\eta(\sigma_t)) = \text{topk}\big(\text{softmax}(\eta(\sigma_t)W_R), k\big) \in \mathbb{R}^N, \tag{28}$$

where $W_R \in \mathbb{R}^{D\times N}$ is the router weight mlp, $\text{softmax}(\cdot)$ is applied along the expert dimension, and $\text{topk}(\cdot, K)$ keeps the K largest entries and sets the others to zero.

Load balancing loss. To prevent expert collapse and encourage all experts to be utilized across noise levels, we add a standard load-balancing loss on the router. For a mini-

Table 6. Hyperparameters

Hyperparameters	Value
Policy Embedding Dimension	768
Transformer Layers	8
Attention Head	8
History Length	2
Action Length	10
Parameter Count (Millions)	628
Sampler	DDIM
Optimizer	AdamW
Beta	[0.9, 0.95]
Learning Rate	1e-4
Language Goal Encoder	clip-vit-large-patch14
Transformer Weight Decay	0.05
Batch Size	128
Train Steps in Thousands	100
Training data	50 demos per task

batch $\mathcal{B}$ of size $|\mathcal{B}|$, let

$$m_n = \frac{1}{|\mathcal{B}|} \sum_{(x_t,\sigma_t)\in\mathcal{B}} \mathbf{1}\big(\mathbf{r}_n(\eta(\sigma_t)) > 0\big)\,, \tag{29}$$

$$p_n = \frac{1}{|\mathcal{B}|} \sum_{(x_t,\sigma_t)\in\mathcal{B}} \big[\mathrm{softmax}(\eta(\sigma_t)W_R)\big]_n, \tag{30}$$

denote, respectively, the fraction of samples routed to expert n and the average router probability assigned to expert n. The load-balancing term is then

$$\mathcal{L}_{\mathrm{LB}} = \gamma\, N \sum_{n=1}^{N} m_n\, p_n, \tag{31}$$

with γ a small balancing coefficient (e.g., $\gamma = 0.01$). The final training loss for the MoE denoiser is the EDM reconstruction loss plus $\mathcal{L}_{\mathrm{LB}}$. The network hyperparameters are shown in Tab. 6.

C. Disentangling improvements (Pretraining vs Policy).

We conducted decoupling experiments with 8 NVIDIA A100 GPUs. Tab. 7**(E.1)**Pretraining effectiveness: We fixed our policy and the Q-Former with 32 queries but substituted the visual backbone with SOTA embodied domain visual backbones. All baselines use ViT-L as the starting point and are finetuned on robot specific datasets: MVP, VC-1 are MAE based; SPA is neural rendering based. **(E.2)**Policy Superiority: Keeping our pretrained representations and the Q-Former as connections, we compared our policy against modified RDT* with transformer layer counts reduced to match the same parameter scale (400M, action chunk size = 10, denoising step = 10). Our policy outperformed RDT* under our pretrained backbone, validating that our policy network provides a superior action learning manifold. Besides, to validate the causal attention in policy network, we conduct additional ablations on long horizon tasks on libero long benchmark in Tab. 7**(E.3)**.

Table 7. Ablation studies disentangling contributions.

S.R.	(E.1)				(E.2)		(E.3)	
	MVP [52]	VC-1 [33]	SPA [64]	Ours	RDT* [30]	Ours	Full Attn.	Causal Attn.
Libero-10	56.6	64.0	79.3	**89.2**	81.3	**89.2**	82.7	**89.2**

D. Action Throughput Results.

To evaluate the deployment viability of our method, we conduct a benchmark of action throughput on a single NVIDIA A100 GPU. As illustrated in Fig. 9, our model achieves a significant leap in inference speed, reaching a throughput of 124.7 Hz for 7-dimensional action generation. In comparison, the representative VLA baseline, OpenVLA [25], only yields 4.2 Hz, primarily due to the heavy computational overhead of its large language model backbone. Notably, our method is 29.7× faster than OpenVLA and substantially outperforms the competitive diffusion-based baseline RDT* (77.5 Hz). This high-frequency performance demonstrates that our architecture effectively mitigates the inference bottleneck, making it highly suitable for real-time, low-latency robotic control in dynamic environments.

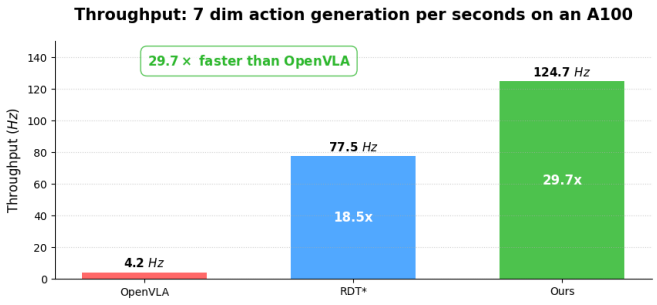

Figure 9. Action throughput on The Libero Benchmark..