Planning in 8 Tokens: A Compact Discrete Tokenizer for Latent World Model

Supplementary Material

This supplementary material provides detailed implementation specifics and additional experimental results for CompACT. We organize the content as follows:

- **Sec. A** provides theoretical framework for the planning sufficiency of compact tokens.
- **Sec. B** presents cross-backbone ablation for CompACT.
- **Sec. C** provides a planning latency breakdown.
- **Sec. D** evaluates closed-loop robot arm manipulation on Robomimic [22].
- **Sec. E** describes the complete CompACT tokenizer architecture, training procedure, and hyperparameters.
- **Sec. F** details the Inverse Dynamics Model (IDM) used for action prediction experiments on RoboNet [8].
- **Sec. G** presents the world model architectures for both navigation (autoregressive with fixed history window) and manipulation tasks (block-causal parallel prediction).
- **Sec. H** explains the Cross-Entropy Method used for navigation planning.
- **Sec. I** provides additional qualitative results including reconstruction examples, attention visualizations, and planning rollouts.
- **Sec. J** provides comparison between tokenizers in terms of planning efficiency.
- **Sec. K** presents that with compact latent tokens we can scale up the world model while remaining efficient.

A. Planning sufficiency of compact tokens

How many bits must a latent representation $\boldsymbol{z}$ carry so that planning in latent space is equivalent to planning in observation space? We formalize this question using mutual information and derive a tight lower bound on the entropy of any planning-sufficient representation $\boldsymbol{z}$. Let $\boldsymbol{o}$ denote an observation, $\boldsymbol{a}^*$ denote the optimal action (or action sequence), and $\boldsymbol{z} = \mathcal{E}(\boldsymbol{o})$ denote the latent representation.

Definition 1 (Planning sufficiency)**.** *A latent representation $\boldsymbol{z} = \mathcal{E}(\boldsymbol{o})$ is* planning-sufficient *if, for a optimal planning algorithm π, planning in latent space yields the same optimal action as planning in observation space: $\pi(\boldsymbol{z}) = \pi(\boldsymbol{o})$. This requires that $\boldsymbol{z}$ retains all action-relevant information from $\boldsymbol{o}$, i.e.,*

$$I(\boldsymbol{z}; \boldsymbol{a}^*) \;=\; I(\boldsymbol{o}; \boldsymbol{a}^*). \tag{1}$$

Proposition 1 (Minimum description length for planning)**.** *If the optimal planning algorithm π is deterministic, i.e., $H(\boldsymbol{a}^* \mid \boldsymbol{o}) = 0$, then a planning-sufficient representation $\boldsymbol{z}$ (Def. 1) exists with minimum entropy*

$$\min_{\boldsymbol{z}:\, \textit{plan-suff.}} H(\boldsymbol{z}) \;=\; I(\boldsymbol{o}; \boldsymbol{a}^*) \;=\; H(\boldsymbol{a}^*) \;\ll\; H(\boldsymbol{o}), \tag{2}$$

established by necessity (no planning-sufficient $\boldsymbol{z}$ can have lower entropy) and achievability (a $\boldsymbol{z}$ attaining this bound exists).

Proof.

(a) (Necessity) By the definition of mutual information, $I(\boldsymbol{z}; \boldsymbol{a}^*) = H(\boldsymbol{z}) - H(\boldsymbol{z} \mid \boldsymbol{a}^*) \leq H(\boldsymbol{z})$. Combining this with the planning-sufficiency condition $I(\boldsymbol{z}; \boldsymbol{a}^*) = I(\boldsymbol{o}; \boldsymbol{a}^*)$ (Eq. 1) yields $H(\boldsymbol{z}) \geq I(\boldsymbol{o}; \boldsymbol{a}^*)$.

(b) (Achievability) When $H(\boldsymbol{a}^* \mid \boldsymbol{o}) = 0$, define the deterministic encoder $\mathcal{E}(\boldsymbol{o}) := \pi^*(\boldsymbol{o}) = \boldsymbol{a}^*$. Then $I(\boldsymbol{z}; \boldsymbol{a}^*) = H(\boldsymbol{a}^*) = I(\boldsymbol{o}; \boldsymbol{a}^*)$, so $\boldsymbol{z}$ is planning-sufficient, and $H(\boldsymbol{z}) = H(\boldsymbol{a}^*) = I(\boldsymbol{o}; \boldsymbol{a}^*)$, achieving the bound. □

Remark. The deterministic assumption $H(\boldsymbol{a}^* \mid \boldsymbol{o}) = 0$ refers to the existence of a unique optimal action for each observation, not to the planning procedure itself. In practice, CompACT uses the Cross-Entropy Method (CEM), which employs stochastic sampling to search for this optimum; the stochasticity is in the optimizer, not in the underlying observation-to-action mapping.

Connection to CompACT. In robotic manipulation and navigation, action spaces are low-dimensional (e.g., 3–5 DOF), so $H(\boldsymbol{a}^*) \ll H(\boldsymbol{o})$. Proposition 1 implies that a planning-sufficient tokenizer need not preserve the full observation entropy as in conventional autoencoder; the information-theoretic floor is at most $H(\boldsymbol{a}^*)$ bits. CompACT encodes each frame into 8–16 discrete tokens using FSQ with levels $[8, 8, 8, 5, 5, 5]$, yielding $\approx 2^{16}$ codes per token ($\approx$16 bits/token), for a total of 128–256 bits per frame. Considering that previous work leveraged $\approx$300–400 bits to encode the whole image [20, 32] and $H(\boldsymbol{a}^*) \ll H(\boldsymbol{o})$, this is expected to far exceed the action-entropy bound, providing a comfortable margin for planning sufficiency. Empirically, Tab. 3 of the main confirms this: an inverse dynamics model trained on CompACT tokens (16 tokens, $R^2 = 0.716$) outperforms one trained on the baseline tokenizer using $16\times$ more tokens (256 tokens, $R^2 = 0.684$), indicating that the compact representation retains and even enhances the action-relevant information.

B. Cross-backbone ablation for CompACT

The main paper presents CompACT with DINOv3 [28] as the frozen encoder backbone. To verify that our approach does not depend on properties specific to DINOv3, we replace it with two architecturally distinct vision foundation models (VFMs)—MAE [13], trained via masked image modeling,

Table 1. **Cross-backbone ablation of CompACT.** Reconstruction performance of CompACT with 16 tokens on ImageNet validation split across different backbone choices.

Backbone	SigLIP-2	MAE	DINOv3
rFID↓	**2.09**	3.43	2.40

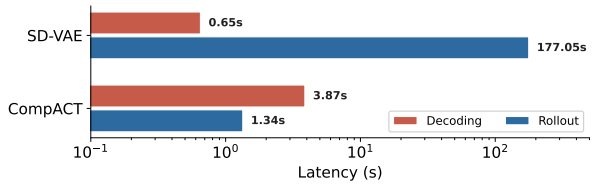

Figure 1. **Planning latency breakdown.** Comparison of rollout and decoding latency between SD-VAE and CompACT on a single RTX 6000 ADA GPU. Note the log scale on the x-axis.

and SigLIP-2 [30], trained via vision-language pretraining—while keeping all other components unchanged. As shown in Tab. 1, all three backbones yield competitive rFID, with SigLIP-2 even surpassing DINOv3. This result confirms that CompACT is not tied to any particular backbone's latent space; rather, it leverages the high-level semantic structure shared across strong VFMs—the planning-relevant information our tokenizer is designed to preserve.

C. Planning latency breakdown

Fig. 1 two major components of the planning latency for a single trajectory optimization into two components: world model *rollout* (forward passes through the world model) and *decoding* (converting latent tokens back to images for cost evaluation). The rollout cost dominates SD-VAE's total planning time (177.05s out of 177.70s), as the world model must process 784 tokens per frame across 1,920 forward passes during MPC. CompACT reduces this bottleneck by 99.2% (177.05s → 1.34s) by operating on only 16 discrete tokens per frame. While our generative decoder is ∼6× slower than SD-VAE's single-step decoder (3.87s vs. 0.65s), decoding accounts for only a small fraction of the total planning budget and is invoked far fewer times than rollout (240 vs. 1,920 calls). Furthermore, when using latent-space cost functions (Tab. 5 in the main paper), decoding is bypassed entirely, yielding even greater speedups.

D. Closed-loop robot arm manipulation

To evaluate CompACT beyond open-loop video prediction, we conduct a preliminary closed-loop manipulation experiment on the Lift task from RoboMimic [22]. We adopt a hierarchical formulation: the world model plans in CompACT's latent space, and the IDM decodes inter-frame actions for execution. For detailed implementation, we refer to Sec. G.2 and Sec. F, respectively.

Table 2. **Closed-loop manipulation on RoboMimic Lift.**

Tokenizer	SR ↑	Steps ↓
Target tokenizer ($\mathcal{D}_\psi \circ \mathcal{E}_\psi$) [4]	56%	66.8
CompACT (ours)	**56%**	**55.1**

Table 3. **Training hyperparameters for CompACT.**

hparams	CompACT
optimizer	AdamW
β_1	0.9
β_2	0.999
weight decay	0.01
lr	0.0001
lr scheduling	cosine
lr warmup steps	10K
batch size (ImageNet)	512
training steps (ImageNet)	500K
training steps (RoboNet finetuning [8])	256
training steps (RoboNet finetuning [8])	100K

Table 4. **Model architecture hyperparameters for CompACT.**

hparams	Latent resampler	$\mathcal{D}_{\text{compact}}$
depth	5	16
dim	768	1024
MLP dim	3072	4096
heads	8	8

As shown in Tab. 2, CompACT (16 tokens) matches the target tokenizer (256 tokens) in success rate while requiring 17% fewer action steps, suggesting that semantic-level latent plans yield more direct paths to the goal. This result provides preliminary evidence that planning-sufficient compression extends to contact-rich manipulation tasks. We note this is a minimal setup; incorporating real-time observations and proprioception into the IDM could further improve performance, which we leave for future work.

E. Details of CompACT tokenizer

Tab. 3 and Tab. 4 summarize the training and model architecture hyperparameters of CompACT, respectively.

Tokenizer architecture. We use frozen DINOv3-B [28] in the encoder $\mathcal{E}_{\text{compact}}$. In DINOv3-B, we re-initialized the last layer normalization's affine parameters (weight and bias to 1 and 0, respectively). We found that using pretrained affine parameters as-is results in codebook collapse during training, since the output of $\mathcal{E}_{\text{compact}}$ has specific statistics when using pretrained layernorm affine parameters. For the latent resampler, we use the 5 transformer decoder blocks similar to DETR [3] and Perceiver [17]. Number of learnable queries determines the number of latent tokens used in CompACT.

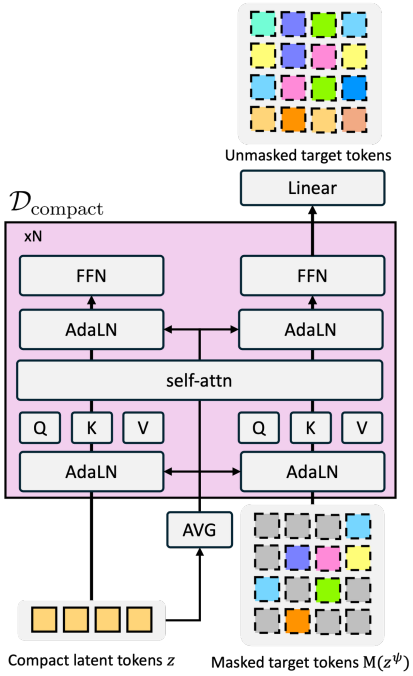

Figure 2. $\mathcal{D}_{\text{compact}}$ **architecture.** The proposed $\mathcal{D}_{\text{compact}}$ is based on MM-DiT [11], a DiT [25] variant designed for multimodal input processing. The architecture consists of two parallel processing streams: one for compact latent tokens $\boldsymbol{z}$ and another for target latent tokens $\boldsymbol{z}^{\psi}$, which are fused through self attention over the concatenated token sequence. For the overall depiction of CompACT, we refer to Fig. 2 in the main paper.

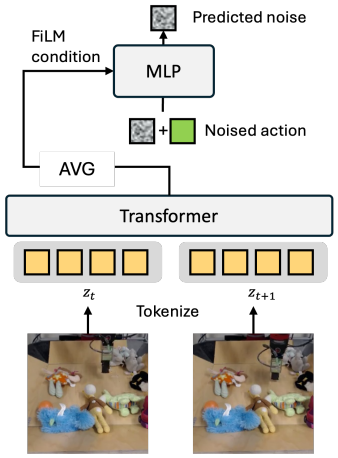

Figure 3. **Inverse Dynamics Model (IDM) architecture.** Consecutive frames are tokenized and processed through a transformer-based frame encoder, which produces a single conditioning vector via average pooling. This vector conditions an action denoiser implemented as a diffusion policy [6], which predicts the action taken between the two frames.

Table 5. **Model architecture hyperparameters for IDM.**

hparams	Frame encoder	hparams	Action denoiser
depth	4	# linear	4
dim	512	hidden dim	512
MLP dim	2048	diffusion	DDPM [16]
heads	8	timesteps	1000
#params	13.5M	#params	3.6M

We use the Finite Scalar Quantization [23] (FSQ) for discretizing the output of latent resampler. Levels per channel are set to $[8, 8, 8, 5, 5, 5]$, which is the recommended level configuration to construct approximately 2^{16} size codebook. For decoder $\mathcal{D}_{\text{compact}}$, we use the MM-DiT [11], which takes compact latent as a condition. Fig. 2 depicts the details architecture of the decoder $\mathcal{D}_{\text{compact}}$, which is shown in simplified form in Fig. 2 in the main paper. $\mathcal{D}_{\text{compact}}$ is trained from the scratch. Proposed CompACT comprises 775M parameters in total, including $\mathcal{E}_{\text{compact}}$, $\mathcal{D}_{\text{compact}}$, and $\mathcal{D}_{\psi}$.

Masked generative modeling. As noted in Sec. 3.2.2 of the main paper, $\mathcal{D}_{\text{compact}}$ is formulated as a masked generative model. During training, we randomly sample the mask ratio from $(0, 1]$ following the cosine masking schedule [4]. During inference, we decode target latent tokens z^{ψ} by progressively unmasking them from a fully masked sequence, using compact latent tokens z as conditioning. In each iteration, a fraction of predictions with high confidence is accepted, while remaining tokens are re-masked. We follow the same cosine masking schedule during inference.

Dataset. CompACT tokenizer is trained on ImageNet-1K [10], on 224×224 (for fair comparison in navigation experiment following NWM [2]) and 256×256 resolution. For the augmentation, we used random crop and random horizontal flip. We additionally finetune the tokenizer for the experiment with RoboNet [8] since robot data has a significant domain gap compared to the ImageNet data. For RoboNet finetuning we used center cropped 256×256 resolution images.

Training and inference. For ImageNet [10] pretraining, CompACT is trained for 500K steps with batch size of 512. We use the AdamW [21] with $1e-4$ learning rate. β_1 and β_2 are set to 0.9 and 0.999, respectively. During inference, sampling step in $\mathcal{D}_{\text{compact}}$ is set to 16.

F. Details of inverse dynamics model (IDM)

Fig. 3 and Tab. 5 present the model architecture and hyperparameters of IDM, respectively.

Architecture. The Inverse Dynamics Model (IDM) used for RoboNet experiments consists of two modules: a frame encoder and an action denoiser. Given latent tokens from two consecutive frames, the frame encoder first processes the concatenated token sequence through a 4-layer transformer encoder. The target tokenizer [4] requires processing 512 tokens, while CompACT requires only 32 tokens—a $16\times$ reduction in sequence length. The frame encoder output is average-pooled into a single vector, which serves as the

Table 6. **Model architecture hyperparameters for world model.**

hparams	navigation	manipulation
depth	12	16
dim	768	1024
MLP dim	3072	4096
heads	12	16
# params	243M	270M

conditioning signal for the action denoiser. The action denoiser is a multi-layer perceptron (MLP) with 4 linear layers (hidden dimension 512), SiLU activation [14], and FiLM conditioning [26] between each layer. The denoiser takes a noisy 5-dimensional action and predicts the applied noise. Following diffusion policy [6], we use DDPM [16] with a squared cosine schedule [24].

Dataset. IDM is trained on RoboNet [8]. Frame pairs are randomly sampled at each iteration. Frames are pre-encoded using tokenizer (CompACT or target tokenizer) with 256×256 resolution center-cropped images. 250 episodes are used for test.

Training and inference. IDM is trained for 100K steps with batch size of 128. We use the AdamW [21] with $1e-4$ learning rate. β_1 and β_2 are set to 0.9 and 0.999, respectively. During inference, we use 1000 diffusion timesteps for sampling.

Evaluation. We evaluate IDM performance using two metrics computed on predicted end-effector positions. First, we measure L1 error between the predicted and ground truth end-effector positions. Second, we compute the coefficient of determination (R^2), which measures how well the model explains the variance in end-effector movements. R^2 ranges from 0 to 1, where higher values indicate better prediction. For instance, $R^2 = 0.9$ means that model can explain 90% of the variance in end-effector movements.

G. World model details

G.1. Navigation: autoregressive with fixed history window

For navigation tasks, we adopt an autoregressive formulation where the world model predicts the next latent state z_{t+1} conditioned on a fixed-length history window of past states $\{z_{t-\tau}, \ldots, z_t\}$ and the action a_t. This design follows the Navigation World Models (NWM) [2] framework but operates in our compact discrete latent space with $N \leq 16$ tokens per timestep. Fig. 4 and Tab. 6 present the model architecture and hyperparameters of world model for navigation, respectively.

Architecture. Fig. 4 illustrates the architecture. We employ a DiT-based [25] architecture with multiple transformer blocks, where each block consists of adaptive layer normalization (AdaLN), multi-head self-attention, cross-attention, and feed-forward networks (FFN). The action a_t is first

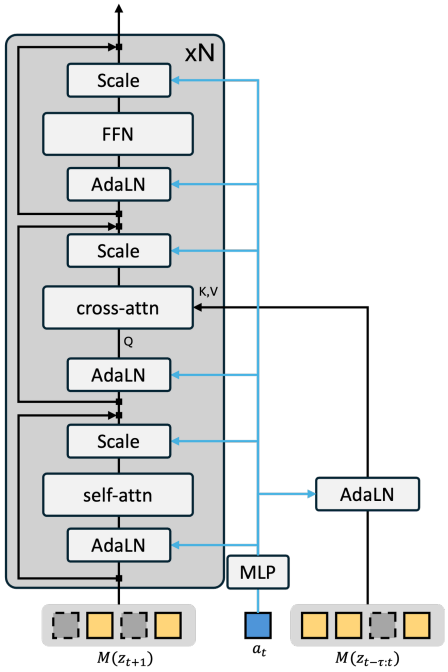

Figure 4. **World model f_ϕ architecture for navigation.** For navigation tasks, the world model is formulated as an action-conditioned generative model that autoregressively generates the next frame. The model follows the CDiT architecture proposed in NWM [2], where actions are conditioned via adaptive layer normalization and history frames are conditioned via cross-attention.

encoded through an MLP and used to modulate the transformer blocks via AdaLN, similar to how DiT conditions on class labels. The model takes as input the masked future tokens $M(z_{t+1})$ and attends to the history window $M(z_{t-\tau:t})$ through cross-attention layers, where $M(\cdot)$ denotes the masking operation. The history tokens serve as keys and values (K, V), while the future tokens act as queries (Q). We use $\tau = 4$ for the history window length and $N = 12$ for the DiT blocks. For SD-VAE experiments, we follow the exact architecture from NWM [2] (CDiT-B). For discrete tokenizers (FlexTok [1] and CompACT), we use the same configuration with two adaptations to handle discrete tokens: (1) embedding and classification layers instead of continuous projections, and (2) masked generative modeling instead of diffusion-based generation.

History masking. Following the principles of diffusion forcing [5], we randomly mask tokens in the history window during training. This encourages the model to learn robust temporal dependencies and improves action conditioning. During each training iteration, we randomly mask tokens in each historical frame. At inference time, we use the slightly masked (20%) history for stable rollouts. Ablation results in Table 5 of the main paper demonstrate that history masking improves planning accuracy.

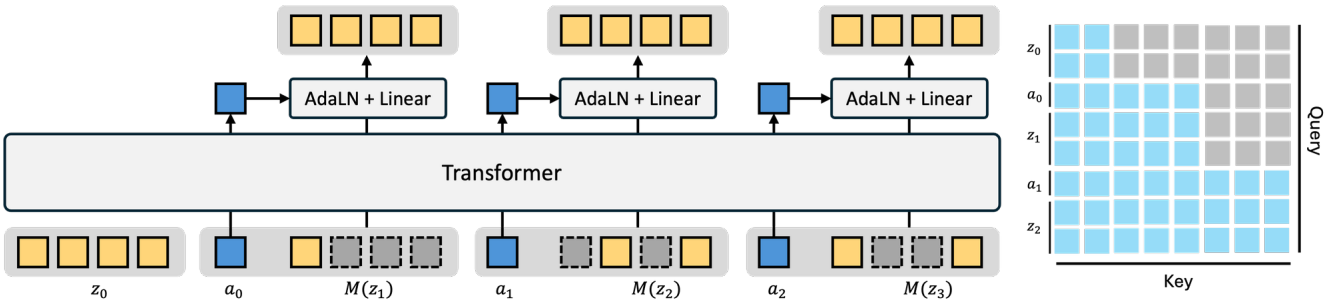

Figure 5. **World model f_ϕ architecture for manipulation.** For manipulation tasks, the world model is formulated as a block-causal transformer that enables parallel prediction of multiple future frames conditioned on historical timesteps. Action tokens are used to condition the prediction head (AdaLN + linear layer) that unmasks latent tokens at each future timestep.

Dataset. The navigation world model is trained on three datasets: RECON [27], SCAND [19], and HuRoN [15]. For HuRoN, we use the publicly available low-resolution version. All datasets contain first-person navigation trajectories with corresponding actions (changes in x-axis, y-axis, and yaw). Frames are center-cropped and resized to 224×224 resolution. We follow the same train/test splits as NWM [2]. We exclude Tartan [31] and Ego4D [12], which were used in the original NWM [2], for the following reasons: Tartan consists predominantly of forward-only actions, and Ego4D, while providing large-scale data, is computationally expensive to process. We found that we can fairly reproduce navigation planning performance without them.

Reproducing NWM [2]. The SD-VAE row in Table 4 of the main paper can be considered as our reproduction of NWM [2]. The original NWM reports ATE of 1.13 and RPE of 0.35 on RECON, while our reproduction achieves ATE of 1.262 and RPE of 0.354. Despite several differences—using a smaller model (CDiT-B instead of CDiT-XL used in the original NWM), low-resolution data for HuRoN, and excluded datasets (Tartan and Ego4D)—we fairly reproduce the navigation planning performance.

Training and inference. The navigation world model is trained for 200K steps with a batch size of 128. We use AdamW optimizer [21] with a learning rate of 1×10^{-4}. We set $\beta_1 = 0.9$ and $\beta_2 = 0.999$, with weight decay of 1×10^{-2}. For the masked generative modeling objective, we randomly sample the mask ratio from $(0, 1]$ following the cosine masking schedule [4]. During inference, we used 8 and 4 sampling steps for 16 and 8 latent tokens, respectively.

Evaluation. We evaluate navigation planning performance using two standard trajectory metrics: Absolute Trajectory Error (ATE) and Relative Pose Error (RPE) [29]. First, ATE measures the RMSE between predicted and ground truth transformations from the initial frame across the entire trajectory between corresponding timesteps. Second, RPE measures the error in relative transformations between consecutive timesteps. Lower values indicate better planning performance for both metrics. For details of model-predictive control framework used for navigation planning, we refer Sec. H.

G.2. Manipulation: block-causal parallel prediction

For robotic manipulation tasks on RoboNet [8], we adopt a block-causal transformer architecture that predicts multiple future frames $\{z_{t+1}, \ldots, z_{t+K}\}$ in parallel, conditioned on the initial observation z_t and action sequence $\{a_t, \ldots, a_{t+K-1}\}$. This parallel formulation is more suitable for video generation tasks where we need to produce extended action-conditioned rollouts efficiently. Fig. 5 and Tab. 6 present the model architecture and hyperparameters of world model for manipulation, respectively.

Architecture. Figure 5 illustrates the block-causal architecture. Unlike the autoregressive model, all future frames are processed simultaneously within a single transformer. The input sequence is constructed by interleaving observations and actions: $[z_{t-\tau:t}, a_t, M(z_{t+1}), \ldots, a_{t+H-1}, M(z_{t+H})]$, where $M(\cdot)$ denotes masked tokens. The key component is the block-causal attention mask (shown on the right of Figure 5): tokens at timestep $t+i$ can attend to all tokens up to and including z_{t+i} and a_{t+i-1}, but cannot attend to future observations or actions. This preserves causal structure while enabling parallel prediction. Each action is encoded through a linear layer. The output token corresponding to each action is then used to condition a prediction head (AdaLN + Linear layer), which produces the unmasked tokens for the subsequent frame. During training, we set the prediction horizon to $H = 14$ (predicting the next 14 frames in parallel). However, the model can generalize to arbitrary horizon lengths since we use causal masking. We provide the first two frames of each episode as context for the model to condition on ($\tau = 2$).

Baseline configuration. For fair comparison, experiments using the target tokenizer (MaskGIT-VQGAN [4]) employ the same block-causal architecture. The only difference

is the sequence length: the target tokenizer processes 256 tokens per frame, while CompACT processes only 16 tokens per frame, resulting in significantly faster generation (Tab. 6 in the main paper).

Diffusion forcing interpretation. This architecture naturally implements causal, discrete extension of diffusion forcing [5]: during training, masked tokens at different timesteps provide varying levels of noisy conditioning to future frames. A frame with fewer masked tokens acts as a cleaner conditioning signal, while heavily masked frames force the model to rely more on learned dynamics and action conditioning. This training scheme improves the model's ability to generate consistent action-conditioned video sequences.

Dataset. The manipulation world model is trained on RoboNet [8], similar to IDM experiment. Frames are pre-encoded using robonet finetuned CompACT with 256 $\times$ 256 resolution center-cropped images. For the target tokenizer [4] baseline, we use it as-is without RoboNet finetuning, which is fair since finetuned CompACT also keeps the frozen target tokenizer. 250 episodes are used for test.

Training and inference. The manipulation world model is trained for 100K steps with a batch size of 128. We use AdamW optimizer [21] with a learning rate of 1×10^{-4}. We set $\beta_1 = 0.9$ and $\beta_2 = 0.999$, with weight decay of 1×10^{-2}. For the masked generative modeling objective, we randomly sample the mask ratio from $(0, 1]$ following the cosine masking schedule [4]. We used 100 sampling steps to generate future 14 frames.

H. Planning with cross-entropy method

As described in Sec. 3.1 and Sec. 3.3, we use a trained world model to standalone-plan goal-conditioned navigation trajectories by optimizing distance between final prediction and the goal. Here, we provide additional details about the optimization using the Cross-Entropy Method (CEM) [7, 9] and the hyperparameters used.

Cross-Entropy Method. For CEM planning, we strictly follow the protocol of NWM [2] for fair comparison. Algo. 1 presents the complete CEM procedure. Specifically, trajectory is assumed to be a straight line and only its endpoint is optimized, represented by three variables: a single translation u and yaw rotation ϕ. This is converted into an action sequence by evenly distributing the translation across H timesteps and applying the rotation only at the final step. Each action step corresponds to a fixed time interval of 0.25 seconds.

Hyperparameters. We set the population size to 80 since increasing over it does not lead to substantial gain in planning accuracy. For other hyperparameters, we follow the configuration of NWM: 2 seconds planning ($H = 8$), single iteration ($I = 1$), top-K selection with $K = 5$, and $M = 3$ repetitive stochastic rollouts per candidate action.

Distance metric $d(\cdot, \cdot)$. The distance between predicted and goal observations can be measured in either pixel space or latent space. For pixel-space evaluation, we decode latent tokens to images using $\mathcal{D}_\psi \circ \mathcal{D}_{\text{compact}}$ and compute the LPIPS distance [18]. Alternatively, we can compute distances directly in the discrete latent space, which offers significant computational speedup by avoiding the decoding step (Tab. 5 (*middle*) in the main paper). This latent-space distance computation is enabled by the properties of Finite Scalar Quantization (FSQ) [23], which we use for discretization. Unlike traditional vector quantization that uses arbitrary codebook indices, FSQ assigns each latent vector to a discrete code based on level-based radix representation, preserving the continuous geometric structure in the discrete space. Specifically, each dimension of the latent is quantized to one of L equally-spaced levels, allowing us to map discrete token indices back to their level-based radix representation. Given two discrete latent tokens $z_i, z_j \in \{1, \ldots, K\}$, we compute their distance as:

$$d(z_i, z_j) = \|\texttt{FSQ}^{-1}(z_i) - \texttt{FSQ}^{-1}(z_j)\|_1, \tag{3}$$

where $\texttt{FSQ}^{-1}(\cdot)$ maps the discrete index to its corresponding level-based radix representation. This distance in the discrete latent space provides a meaningful measure of semantic similarity while enabling faster planning with marginal degradation compared to pixel-space metrics.

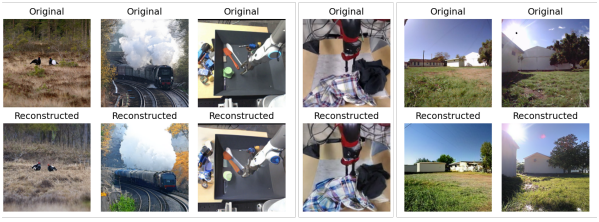

Figure 6. **Qualitative results of reconstruction with CompACT.**

I. More qualitative results

Fig. 6 shows reconstruction examples across ImageNet, RECON, and RoboNet. While CompACT discards fine-grained textures and lighting details due to extreme compression, it preserves semantic content and spatial structure essential for planning tasks. Fig. 7 visualizes attention patterns in the latent resampler, demonstrating that each compact token attends to semantically coherent regions. Fig. 9 presents example planning results with CompACT. While fine-grained details such as textures and shadows are synthesized rather than reconstructed, the rollouts accurately preserve planning-critical information: spatial layout, object positions, and scene structure necessary for effective goal-reaching. Fig. 10 presents additional examples of action-conditioned video generation on RoboNet. Videos generated from CompACT latents maintain more consistent action-driven end-effector movements throughout the rollout compared to the ones generated with target tokenizer latents, validating that the

Algorithm 1 Cross-Entropy Method for Navigation Planning

Require: Initial frame o_0, goal frame o_{goal}, planning horizon H, World model f_ϕ, tokenizer $\mathcal{D}_{\text{compact}} \circ \mathcal{E}_{\text{compact}}$
Require: CEM params: population size N, top samples to be selected K, iterations I, samples per candidate M

// Initialize action distribution
$\mu \leftarrow (\mu_x, \mu_y, \mu_\phi)$
$\Sigma \leftarrow \text{diag}(\sigma_x^2, \sigma_y^2, \sigma_\phi^2)$
Encode observations: $z_0 \leftarrow \mathcal{E}_{\text{compact}}(o_0)$, $z_{\text{goal}} \leftarrow \mathcal{E}_{\text{compact}}(o_{\text{goal}})$
for $i = 1$ to I **do**
  $\mathcal{A} \leftarrow \{(u_j, \phi_j)\}_{j=1}^N$ where $(u_j, \phi_j) \sim \mathcal{N}(\mu, \Sigma)$ // Sample candidate actions
  for each candidate $(u_j, \phi_j) \in \mathcal{A}$ **do**
    // Evaluate via stochastic rollouts
    for $m = 1$ to M **do**
      $z_t^{(m)} \leftarrow z_0$
      $\mathbf{a}^{(j)} \leftarrow \texttt{ToActionSeq}(u_j, \phi_j, H)$
      for $t = 0$ to $H - 1$ **do**
        $z_{t+1}^{(m)} \sim f_\phi(z_t^{(m)}, a_t^{(j)})$ // World model rollout
      end for
      $\hat{o}_H^{(m)} \leftarrow \mathcal{D}_\psi \circ \mathcal{D}_{\text{compact}}(z_H^{(m)})$
      $\hat{o}_{\text{goal}} \leftarrow \mathcal{D}_\psi \circ \mathcal{D}_{\text{compact}}(z_{\text{goal}})$
      $c_m \leftarrow d(\hat{o}_H^{(m)}, \hat{o}_{\text{goal}})$ or $c_m \leftarrow d(z_H^{(m)}, z_{\text{goal}})$// LPIPS between reconstruction or L1 distance between latent
    end for
    $C_j \leftarrow \frac{1}{M} \sum_{m=1}^M c_m$
  end for
  $\tilde{\mathcal{A}} \leftarrow$ top-K from $\mathcal{A}$ with lowest costs $\{C_j\}$ // Select top-K candidates
  // Update distribution parameters
  $\mu \leftarrow \frac{1}{K} \sum_{(u,\phi) \in \tilde{\mathcal{A}}} (u, \phi)$
  $\Sigma \leftarrow \frac{1}{K} \sum_{(u,\phi) \in \tilde{\mathcal{A}}} ((u, \phi) - \mu)((u, \phi) - \mu)^\top$
end for
return Action with minimum cost from $\tilde{\mathcal{A}}$

modular latent tokens effectively capture dynamics-relevant information for manipulation tasks.

J. Planning efficiency analysis

Fig. 8 visualizes the trade-off between planning accuracy (ATE), planning latency, and model size across different tokenizers on RECON. Bubble size represents the peak VRAM usage during planning. CompACT variants achieve superior efficiency: CompACT delivers up to 80× speedup over SD-VAE while maintaining comparable accuracy. In contrast, FlexTok variants suffer significant accuracy loss and large VRAM requirements despite similar token counts, validating that our tokenizer design is critical for enabling efficient real-time planning.

K. Scaling up with fewer tokens

The compact latent representation of CompACT enables scaling world models to larger capacities while maintaining practical planning latency. We train a 750M-parameter variant of our world model for navigation by increasing the depth to 24 layers and hidden dimension to 1024. This scaled model achieves improved planning accuracy with ATE of 1.305 and RPE of 0.370 on RECON—outperforming our base 16-token model (ATE=1.330, RPE=0.390). Planning latency is 24.7 seconds per trajectory, still 7× faster than the SD-VAE baseline (178.78 seconds).

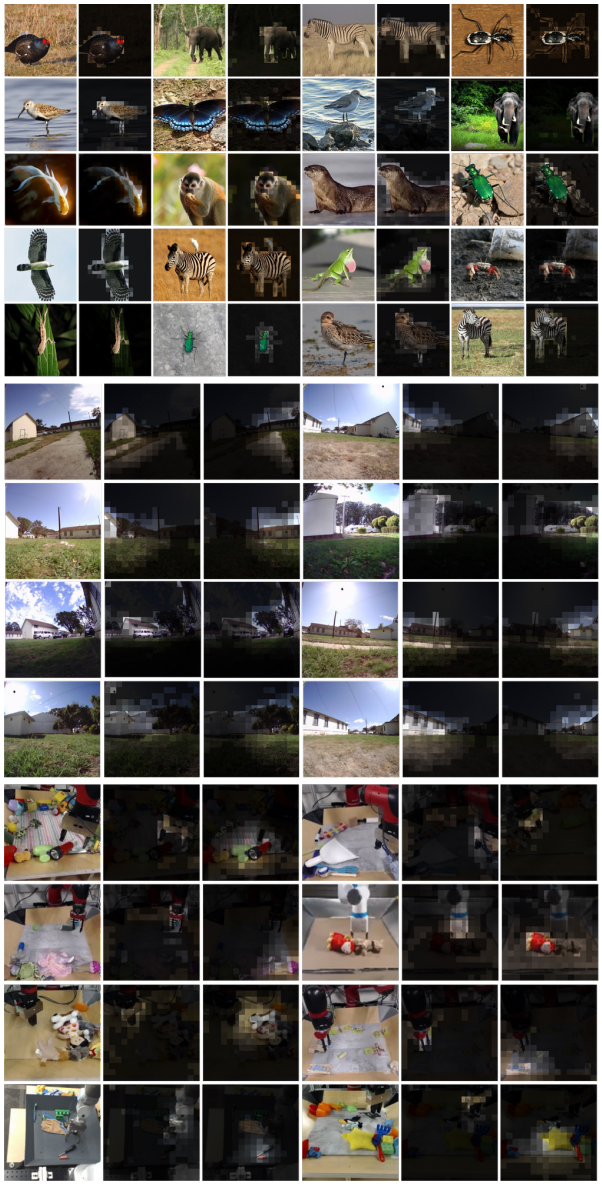

Figure 7. **Attention visualization for compact latent tokens in latent resampler.** Brighter the color, higher the attention score.

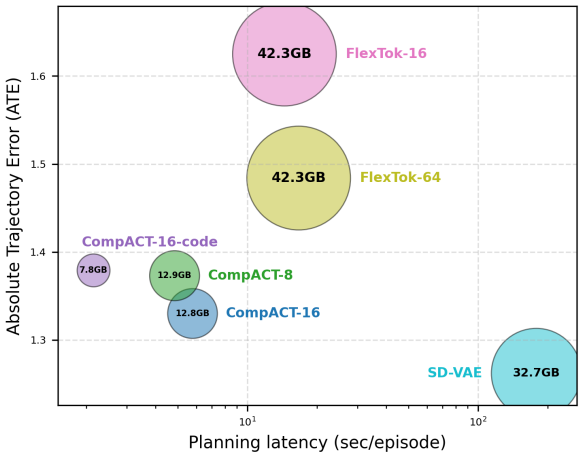

Figure 8. **Plot for ATE, planning latency, and memory peak usage on RECON [27].** Latency and memory usage is mesaured for single trajectory optimization, using a single RTX 6000 ADA GPU.

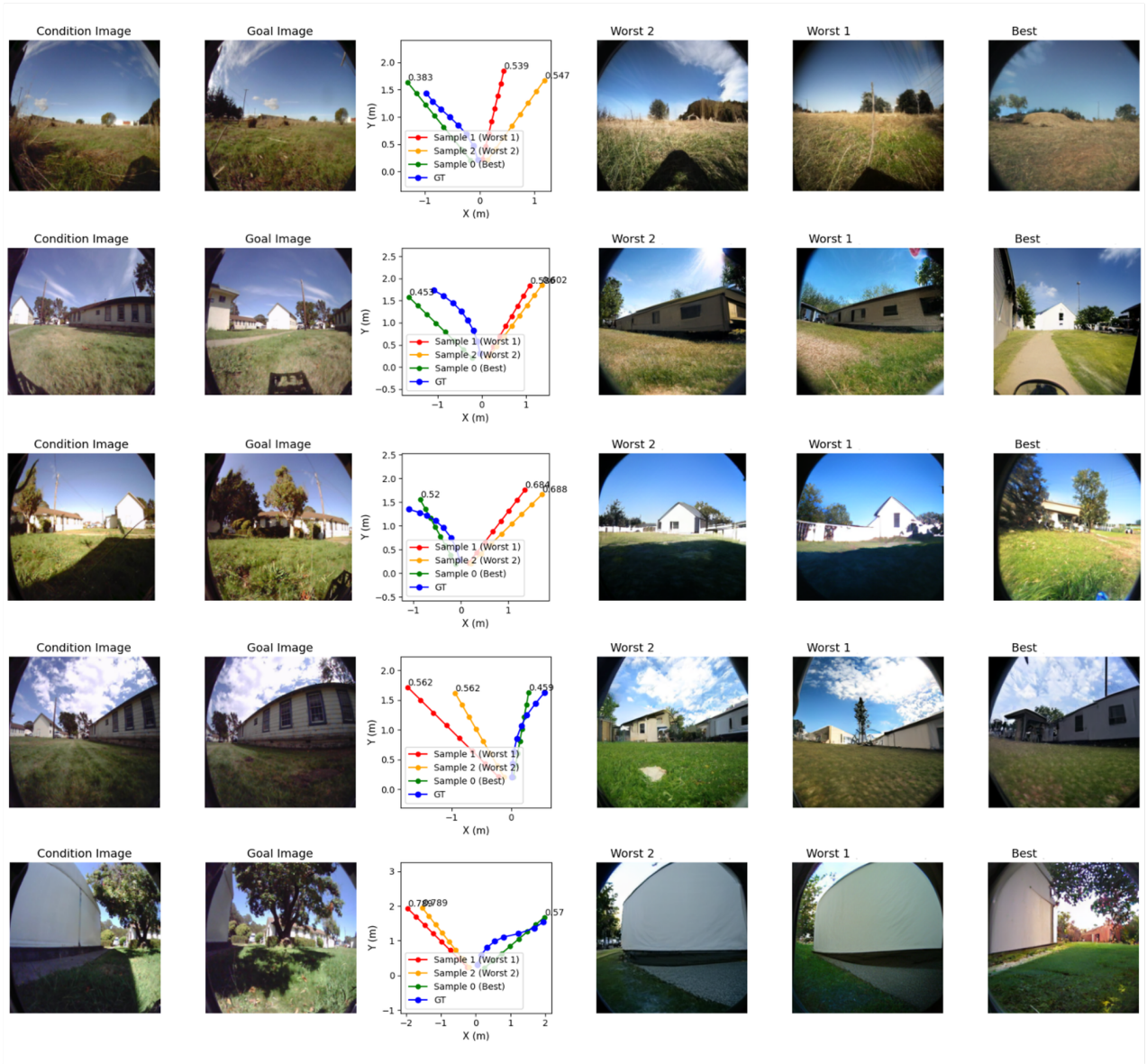

Figure 9. **Additional qualitative results of navigation planning with the proposed CompACT.** Among candidates, worst two action sequences and best action sequences are presented together.

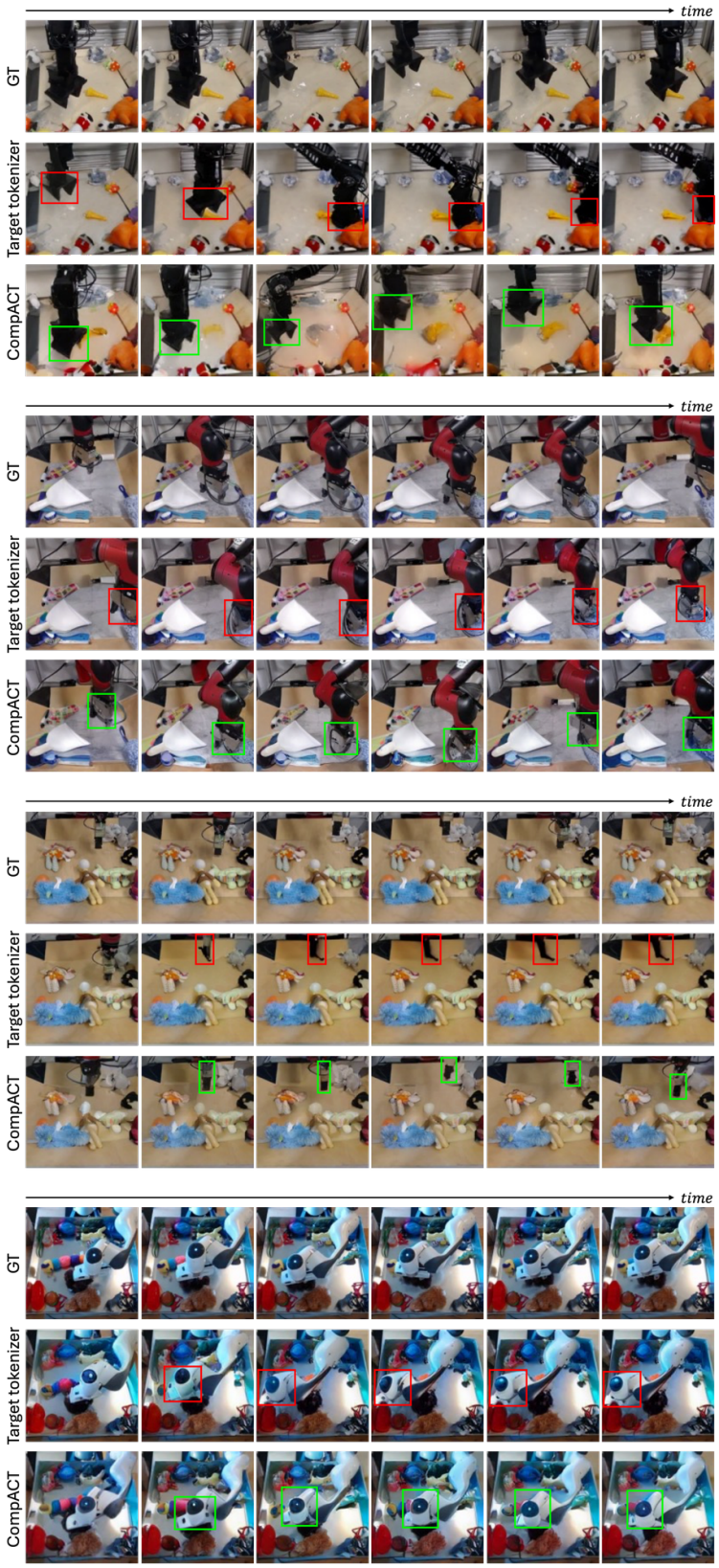

Figure 10. **Additional qualitative results of action-conditioned video generation.** Red and green boxes indicate incorrect and correct end-effector positions, respectively.

References

[1] 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 *Proc. International Conference on Machine Learning (ICML)*, 2025. 4

[2] Amir Bar, Gaoyue Zhou, Danny Tran, Trevor Darrell, and Yann LeCun. Navigation world models. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 3, 4, 5, 6

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Proc. European Conference on Computer Vision (ECCV)*, 2020. 2

[4] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 3, 5, 6

[5] Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. In *Proc. Neural Information Processing Systems (NeurIPS)*, 2024. 4, 6

[6] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023. 3, 4

[7] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Proc. Neural Information Processing Systems (NeurIPS)*, 2018. 6

[8] Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. Robonet: Large-scale multi-robot learning. *Annual Conference on Robot Learning (CoRL)*, 2019. 1, 2, 3, 4, 5, 6

[9] Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005. 6

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 3

[11] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Proc. International Conference on Machine Learning (ICML)*, 2024. 3

[12] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 5

[13] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1

[14] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 4

[15] Noriaki Hirose, Dhruv Shah, Ajay Sridhar, and Sergey Levine. Sacson: Scalable autonomous control for social navigation. *IEEE Robotics and Automation Letters*, 9(1):49–56, 2024. 5

[16] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Proc. Neural Information Processing Systems (NeurIPS)*, 2020. 3, 4

[17] Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *Proc. International Conference on Machine Learning (ICML)*, 2021. 2

[18] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Proc. European Conference on Computer Vision (ECCV)*, 2016. 6

[19] Haresh Karnan, Anirudh Nair, Xuesu Xiao, Garrett Warnell, Sören Pirk, Alexander Toshev, Justin Hart, Joydeep Biswas, and Peter Stone. Socially compliant navigation dataset (scand): A large-scale dataset of demonstrations for social navigation. *IEEE Robotics and Automation Letters*, 7 (4):11807–11814, 2022. 5

[20] Dongwon Kim, Ju He, Qihang Yu, Chenglin Yang, Xiaohui Shen, Suha Kwak, and Liang-Chieh Chen. Democratizing text-to-image masked generative models with compact text-aware one-dimensional tokens. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2025. 1

[21] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proc. International Conference on Learning Representations (ICLR)*, 2019. 3, 4, 5, 6

[22] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Proc. Annual Conference on Robot Learning (CoRL)*, 2021. 1, 2

[23] Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: Vq-vae made simple. In *Proc. International Conference on Learning Representations (ICLR)*, 2024. 3, 6

[24] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *Proc. International Conference on Machine Learning (ICML)*, 2021. 4

[25] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2023. 3, 4

[26] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proc. AAAI Conference on Artificial Intelligence (AAAI)*, 2018. 4

[27] Dhruv Shah, Benjamin Eysenbach, Nicholas Rhinehart, and Sergey Levine. Rapid exploration for open-world navigation with latent goal models. In *Annual Conference on Robot Learning (CoRL)*, 2021. 5, 8

[28] 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. 1, 2

[29] Jürgen Sturm, Wolfram Burgard, and Daniel Cremers. Evaluating egomotion and structure-from-motion approaches using the tum rgb-d benchmark. In *Proc. of the Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RJS International Conference on Intelligent Robot Systems (IROS)*, 2012. 5

[30] Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, Olivier Hénaff, Jeremiah Harmsen, Andreas Steiner, and Xiaohua Zhai. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025. 2

[31] Wenshan Wang, Delong Zhu, Xiangwei Wang, Yaoyu Hu, Yuheng Qiu, Chen Wang, Yafei Hu, Ashish Kapoor, and Sebastian Scherer. Tartanair: A dataset to push the limits of visual slam. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020. 5

[32] Qihang Yu, Mark Weber, Xueqing Deng, Xiaohui Shen, Daniel Cremers, and Liang-Chieh Chen. An image is worth 32 tokens for reconstruction and generation. In *Proc. Neural Information Processing Systems (NeurIPS)*, 2024. 1