World Foundation Model Platform for Physical AI

Cosmos

A. Buynitsky

Jan 24, 2025

Outline

Outline

TLDR (part 1)

Problem: More efficient to train agents virtually which requires virtual platforms
Solution: World Foundation Models (WFMs) that can be pretrained and finetuned similar to LLMs

WFM Goal: predict future observation $\hat{x}_{t+1}$ based on $x_{0:t}$ and c_t

- $x_{0:t}$ sequence of t timesteps of visual observed data
- c_t takes on many forms (actions, trajectories, text descriptions)
- $x_{0:t}$ sequence of t timesteps of visual observed data

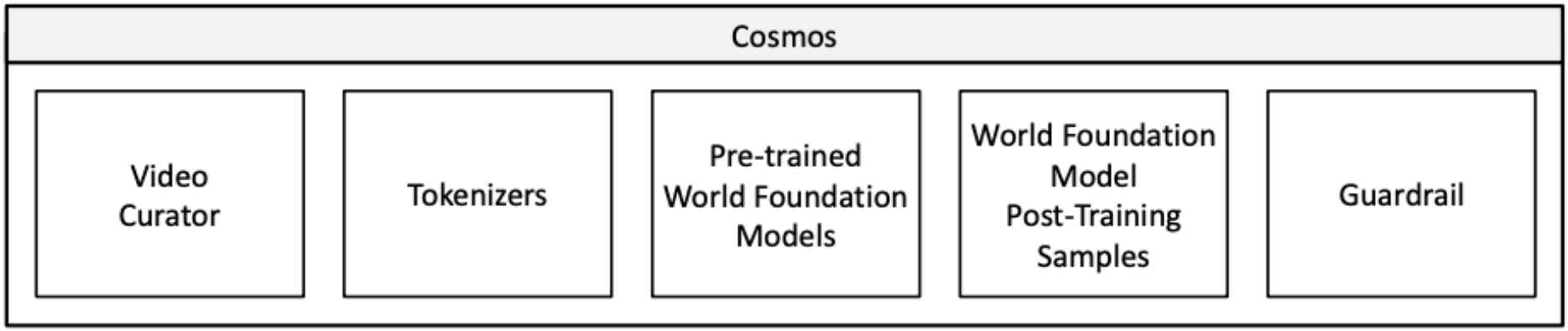
TLDR (part 2)

Problem: More efficient to train agents virtually which requires virtual platforms
Solution: World Foundation Models (WFMs) that can be pretrained and finetuned similar to LLMs
Results:

Research Group: NVIDIA + 1X
Future Work:

- **Policy Evaluation:** benchmarking policies in simulator vs real-world
- **Policy Initialization:** well-trained WFM can initialize policies
- **Policy Training:** WFM pairmed with reward can give feedback in RL
- **MPC Control:** WFM predict future sttates of physical system

TLDR (part 3)



- **Video Currator:** Collecting high-quality pre-training data
- **Tokenizers:** Image and video tokenizers
- **Pre-trained WFMs:** Diffusion and Autoregressive models
- **WFMs Post-Training:** finetuning for specific robotic tasks
- **Guardrail:** safety to block harmful input and outputs

Outline

Raw Data

Collect 20M hours of videos from a diverse set of physical AI environments

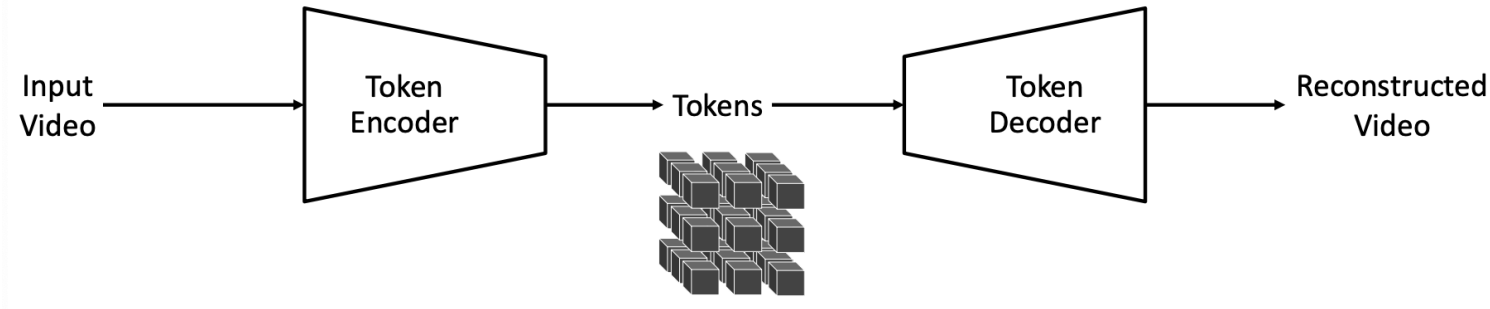
1. Driving (11%)
2. Hand motion and object manipulation (16%)
3. Human motion and activity (10%)
4. Spatial awareness and navigation (16%)
5. First person point-of-view (8%)
6. Nature dynamics (20%)
7. Dynamic camera movements (8%)
8. Synthetically rendered (4%)
9. Others (7%)

Problem: redundant / useless information
Solution: Filtering / Postprocessing
Result: 10^8 pre-training video clips and 10^7 finetuning clips

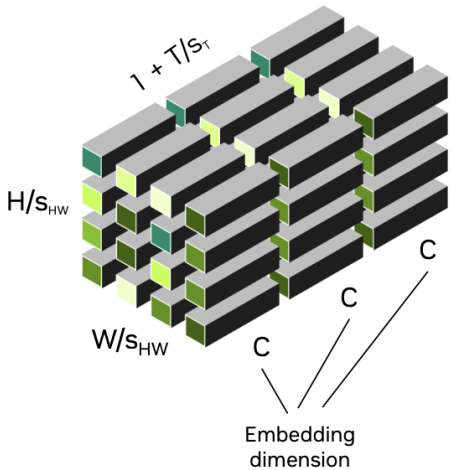
Outline

Tokenizers

Visual tokenizers map raw visual data (images, videos) to compact tokens



Continuous Tokenizers: Encode into continuous latent embeddings



Discrete Tokenizers: Encode into discrete latent codes

S_{HW}: Spatial compression factor, S_T: Temporal compression factor

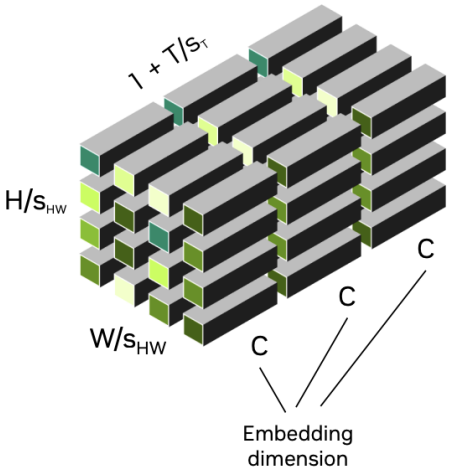
Cosmos Tokenizer

Encoder Decoder Architecture: given $x_{0:T} \in R^{(1+T)\times H\times W\times 3}$

$$\mathcal{E}(x_{0:T}) = z_{0:T'} \in \mathbb{R}^{(1+T')\times H'\times W'\times C'} \tag{1}$$

$$\mathcal{D}(z_{0:T'}) = \hat{x}_{0:T} \in R^{(1+T)\times H\times W\times 3} \tag{2}$$

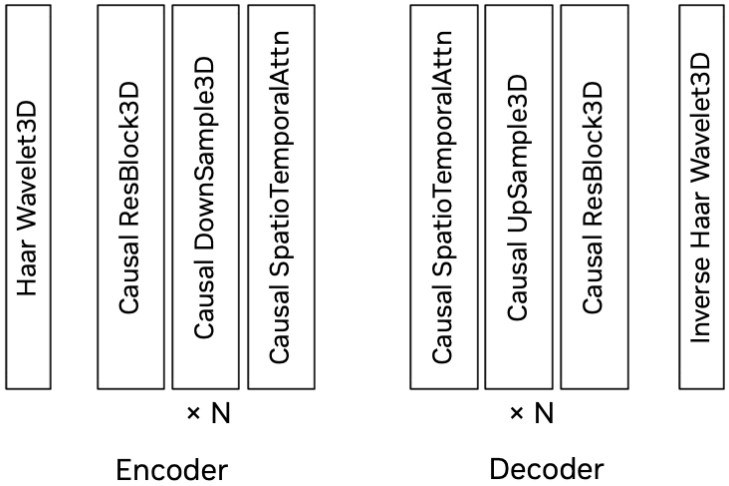
Continuous Tokenizers: Encode into continuous latent embeddings

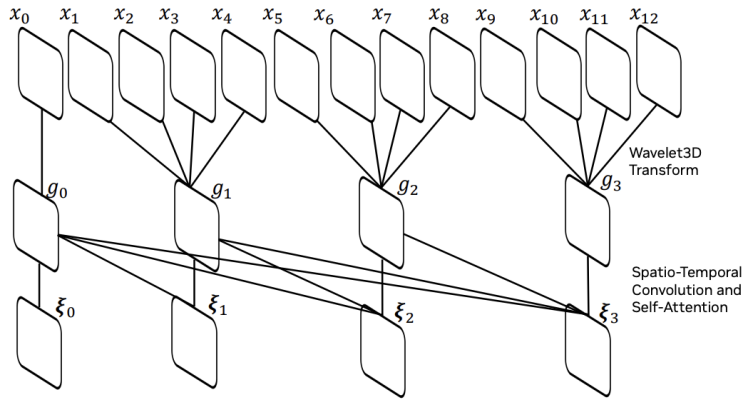


Discrete Tokenizers: Encode into discrete latent codes

S_{HW}: Spatial compression factor, S_T: Temporal compression factor

Cosmos Tokenizer





Wavelet Transform

Wavelet: compress long signals into smaller ones

$$\text{Let } W = \left\{ \begin{vmatrix} 1 \\ 1 \\ 1 \\ 1 \end{vmatrix}, \begin{vmatrix} 1 \\ 1 \\ -1 \\ -1 \end{vmatrix}, \begin{vmatrix} 1 \\ -1 \\ 0 \\ 0 \end{vmatrix}, \begin{vmatrix} 0 \\ 0 \\ 1 \\ -1 \end{vmatrix} \right\} \text{ be the Haar basis}$$

$$E = \{e_1, e_2, e_3, e_4\} \text{ be standard basis of } \mathbb{R}^4$$

(pairwise orthogonal vectors are linearly independent)
The change of basis matrix from W to E, $T_{W,E}$ is given by:

$$T_{W,E} = \begin{vmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & -1 & 0 \\ 1 & -1 & 0 & 1 \\ 1 & -1 & 0 & -1 \end{vmatrix}$$

Wavelet Transform

Given a signal $v = \begin{vmatrix} v_1 & v_2 & v_3 & v_4 \end{vmatrix}^T$, transform v into its coefficients over the Haar Basis by computing $c = \begin{vmatrix} c_1 & c_2 & c_3 & c_4 \end{vmatrix}^T = T_{W,E}^{-1} v$

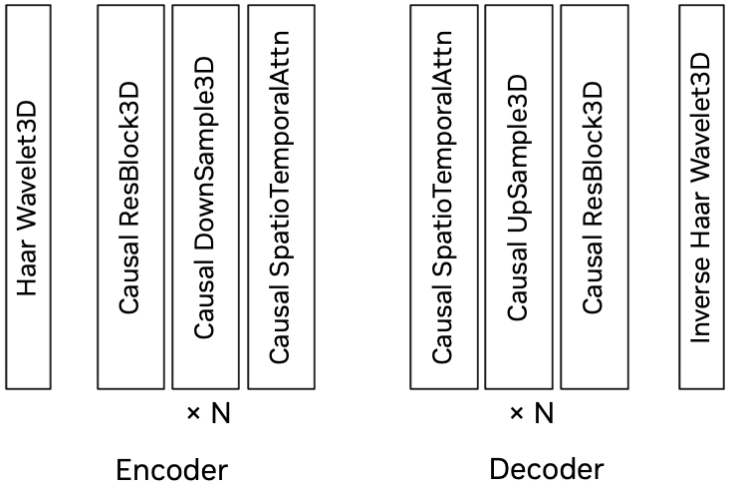
Example: Take $v = \begin{vmatrix} 6 & 4 & 5 & 1 \end{vmatrix}$ over the basis E

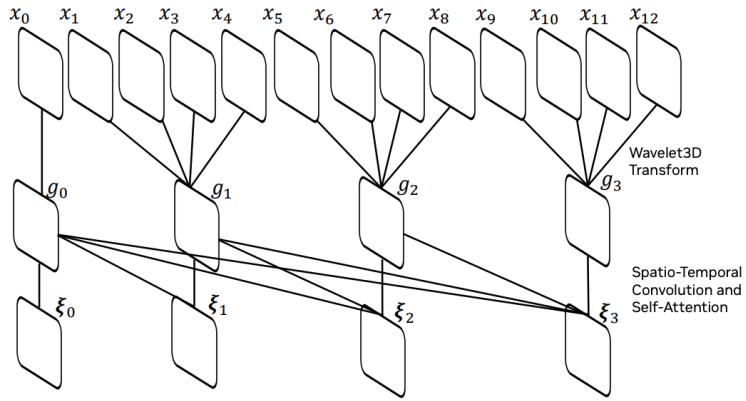
$$c = T_{W,E}^{-1} \cdot v = \begin{vmatrix} 4 & 1 & 1 & 2 \end{vmatrix}$$

Observe that: $c_1 = \frac{v_1+v_2+v_3+v_4}{4}$ is the average value of signal v
c_2 gives coarse details of v
c_3 gives details in first part of v
c_4 gives details in second part of v
Reconstruct signal by $v = T_{W,E} \cdot c$

Compression: Throw away some of coefficients of c (settinig them to zero)
Cosmos Tokenizer does two stages of compression, reducing resultion of x, y and t by half in each (4x reduction total)

Cosmos Tokenizer





Downsampling

- 2D convolution with kernel $1 \times k \times k$ for spatial info
- 1D convolution with kernel $k \times 1 \times 1$ for temporal info (left padded with $k - 1$)
- spatio-temporal attention (ie. $1 + T'$ for last encoder block)

Continuous: Use vanilla Autoencoder for latent space (dim $= 16$)
Discrete: project using Finite-Scalar-Quantization to 64k vocab size

Cosmos Tokenizer Training

Supervise the final output of tokenizer's decoder (no aux loss ie KL), train by alternating images and videos

Stage 1:

$$\mathcal{L}_1 = \|\hat{x}_{0:T} - x_{0:T}\|_1$$

$$\mathcal{L}_{\text{Perceptual}} = \frac{1}{L}\sum_{l=1}^{L}\sum_{t}\alpha_l\|\text{VGG}_l(\hat{x}_t) - \text{VGG}_l(x_t)\|_1$$

Stage 2:

Optical Flow loss for temporal smoothness

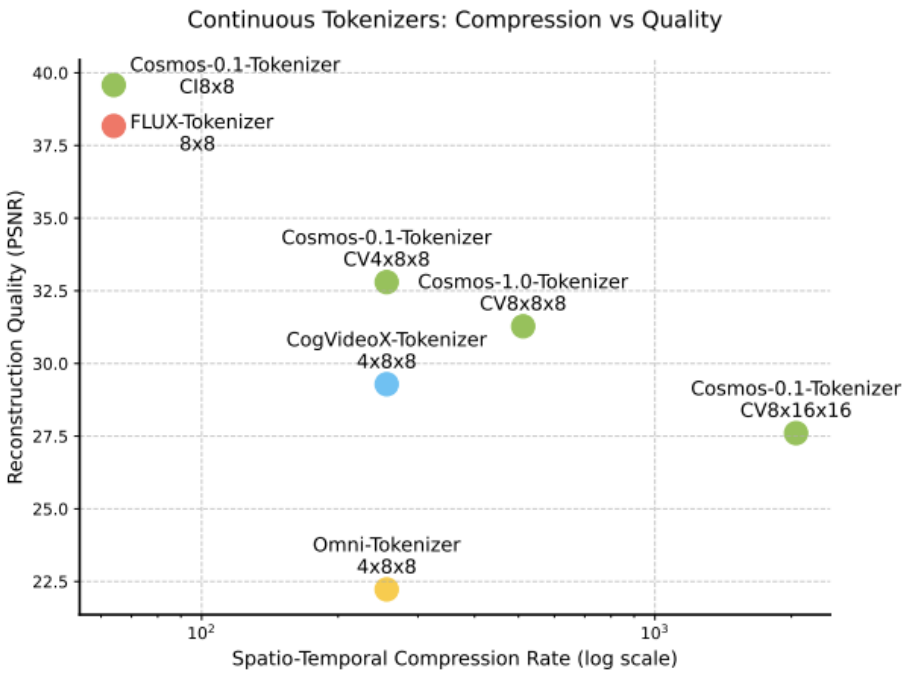
$$\mathcal{L}_{\text{Flow}} = \frac{1}{T}\sum_{t=1}^{T}\|\text{OF}(\hat{x}_t, \hat{x}_{t-1}) - \text{OF}(x_t, x_{t-1})\|_1$$

Gram-matrix loss for sharpness of reconstructed images

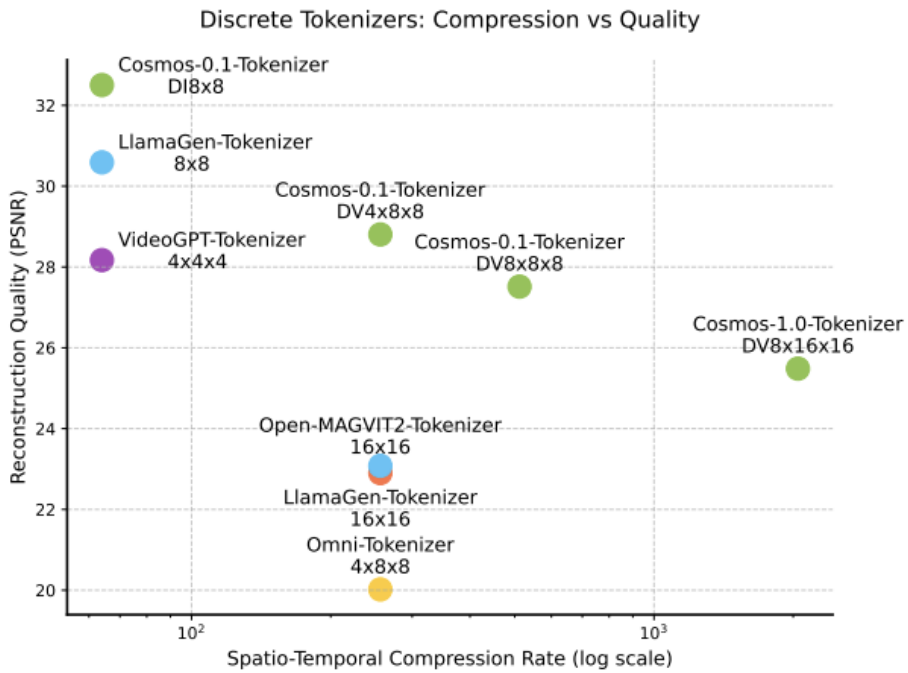
$$\mathcal{L}_{\text{Gram}} = \frac{1}{L}\sum_{l=1}^{L}\sum_{t}\alpha_l\|\text{GM}_l(\hat{x}_t) - \text{GM}_l(x_t)\|_1$$

Results

Better performance than other tokenizers for video data, 2x -12x faster



(a) Continuous tokenizers

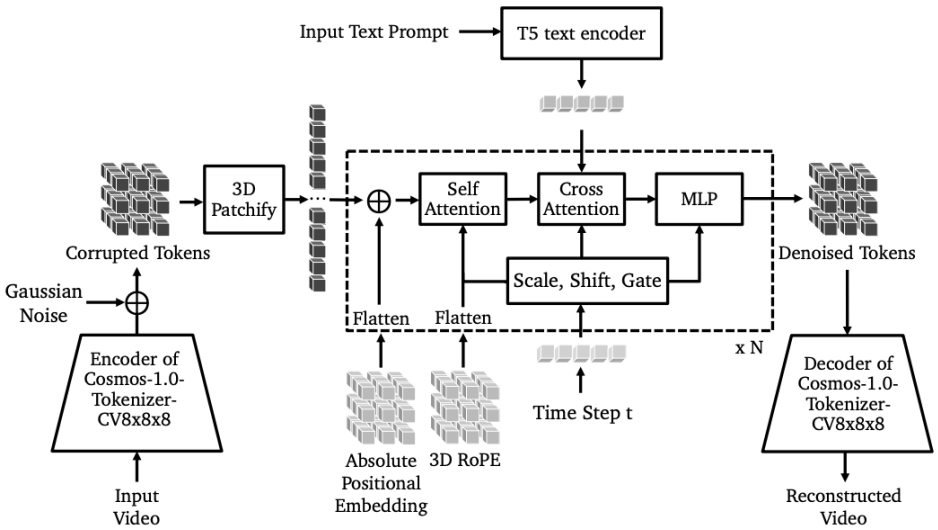


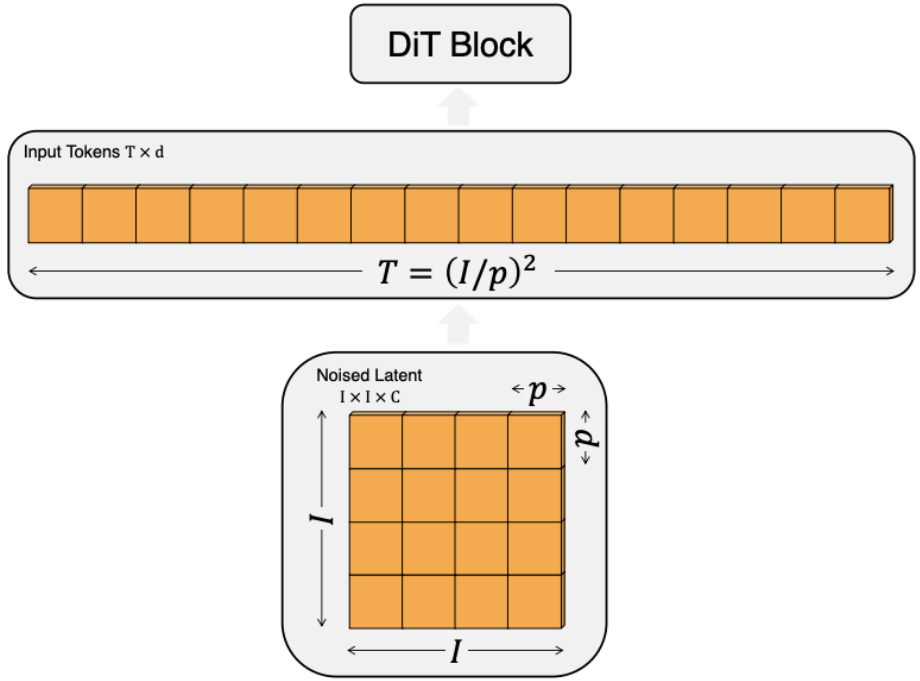
(b) Discrete tokenizers

PSNR (Peak Signal-to-Noise Ratio): compare "true" pixel values of original image to degraded image.

Outline

Architecture (input & 3D Patchify)

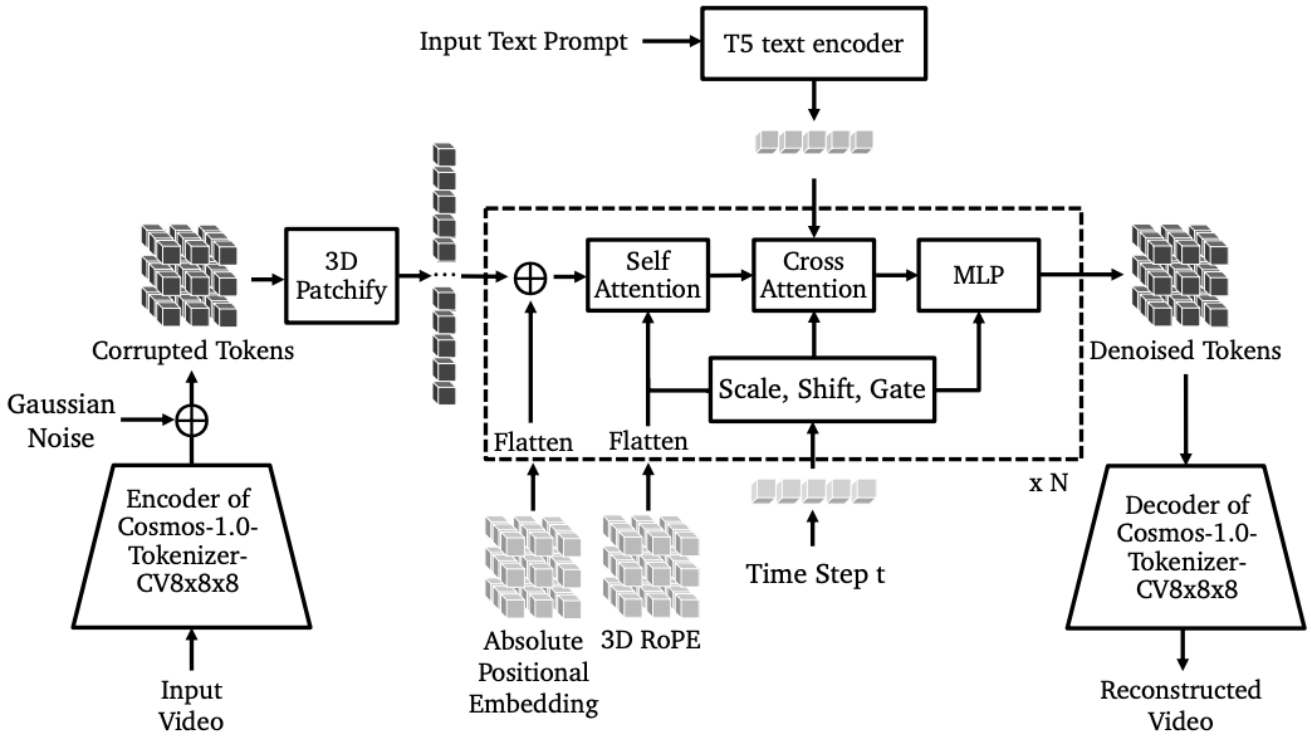




3D Patchification:

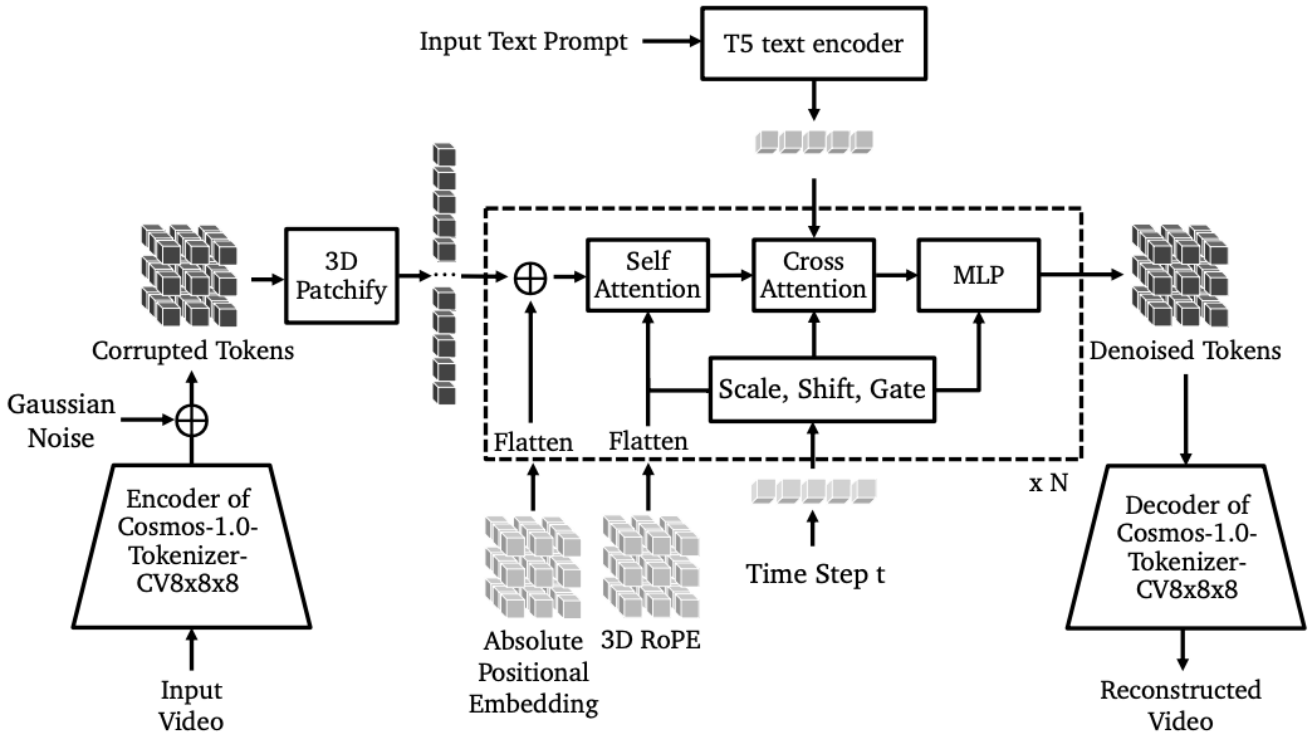
- From tokenizer, input is latent representation $x \in R^{T \times C \times H \times W}$
- Downsample cubes of shape p_t, p_h, p_w into individual input tokens via linear layer and flatten to 1D vector
- image / videos reshaped to 1D, spatiotemporal seq length $\frac{THW}{p_t p_h p_w}$

Architecture (positional encoding)



- partition feature dim into 3 approx equal chunks representing T, W, H, apply RoPE across each dimension
- Rescale temporal frequencies based on video FPS
- Add an absolute (learnable) positional embedding

Architecture (Attention)



Cross Attention: keys and querries from T5-XXL etext encoder

Query-Key Normalization before attention (RMSNorm)

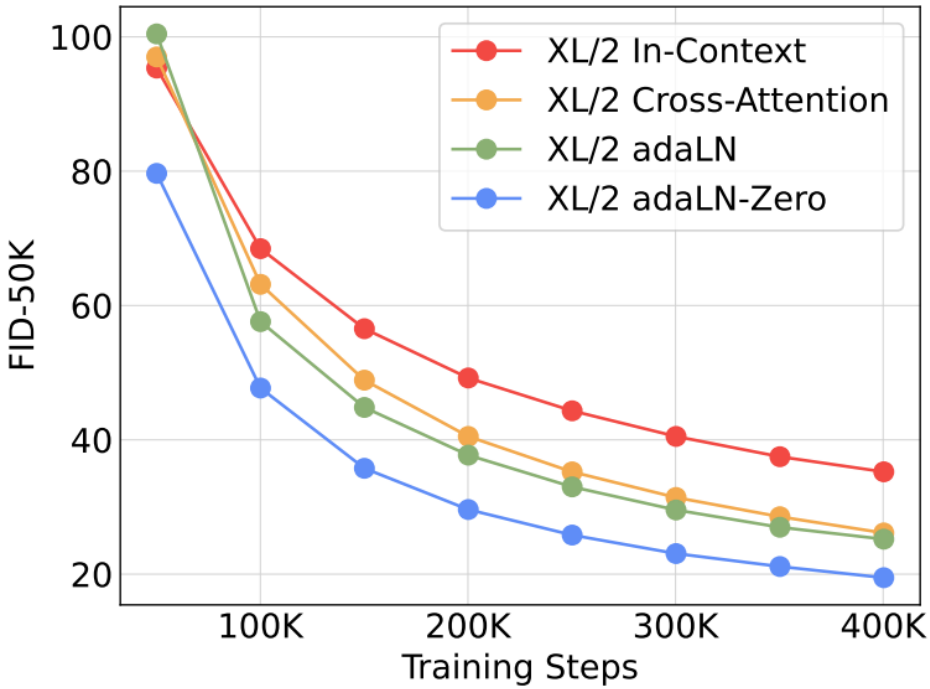
- QK-norm applied before attn layers
- focus on re-scaling, ignore re-centering aspect of normalization
- Significantly reduce computation

Conditioning Transformer Inputs

How to condition transformer on time step t, class label c, prompt ...?

In-context Conditioning:
Append embeddings of t and c as two additional tokens in input sequence. Remove them after final block. $\sim$*0%* $\uparrow$ *Gflops*.

Cross-attention Block
Create sequence of length 2: $[t, c]$. Modify DiT block to have additional mha block *15%* $\uparrow$ *Gflops*.



Is there a more efficient way to achieve performance of cross-attn with less compute?
YES: **Adaptive Layer Normalization (AdaLN)**

Conditioning Transformer Inputs

Layer Norm:

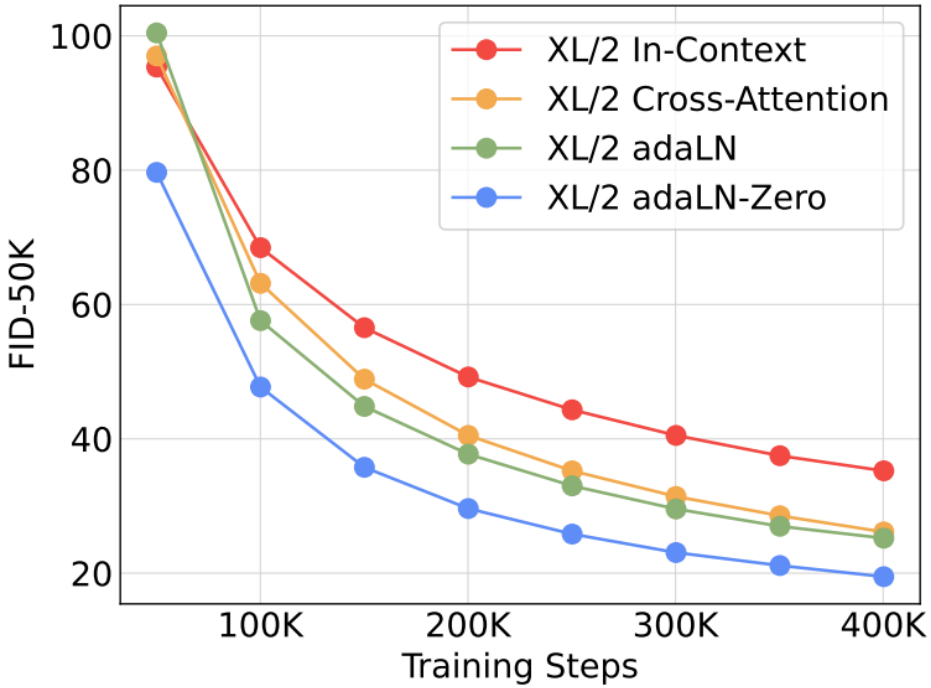
$$\mathbf{h} = \gamma \dot{N}(x) + \beta, N(x) = \frac{x - \mu}{\sigma}$$

μ and σ are mean and sd of x. γ and β are learnable vectors.

Adaptive Layer Normalization (AdaLN):

Learn γ and β from sum of t and c embeddings.

$$\gamma, \beta = \text{MLP}(t + c)$$

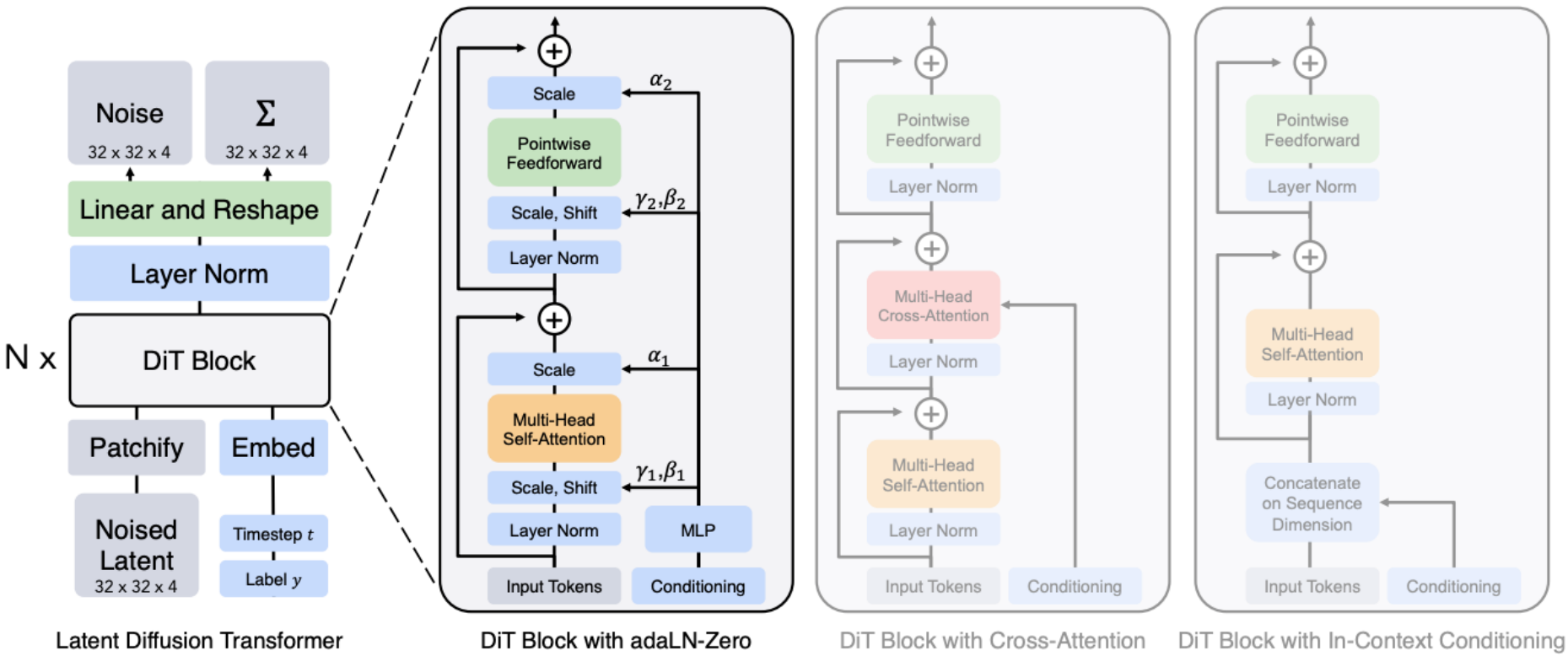


AdaLN-Zero: Inspiration from initializing scalars (residual block as identity fn)

Compute α in addition to γ and β as scaling parameters.

α initialized as 0.

DiT Block



AdaLN-LoRA:

Add LoRA to AdaLN layers, achieving 36% reduction in parameter count, while maintaining performance.

Training Details

Text2World

- Interleave batches of image and video data, scaling from 512p @ 57 frames to 720p @121 frames
- Organize data into 5 aspect ratios (1:1, 3:4,. . .), process frames from same bucket in each batch
- Two copies of model weights: BF16 for forward and backward pass. FP32 used for parameter updates.

Video2World

- Image and Video conditioning by concat. conditional frames f_i with generated frames g_j along temporal dimension
- binary mask concatenated along channel dimension to remove f_i from loss calculation

Prompt Upsampler:

WFM uses detailed video descriptions during training as text inputs.
Finetune LLM to add detail to user prompts during inference

Cosmos Results

30-second videos generated auto-regressively conditioned on first 9 frames.

Prompt: The video depicts the interior of a large industrial facility, likely a factory or warehouse. The space is expansive with high ceilings and metal framework. Overhead cranes and various machinery are visible, indicating a setting for heavy manufacturing or assembly. The floor is mostly empty, with some scattered debris and marked lines. Safety signs and barriers are present, emphasizing the industrial environment. The lighting is natural, streaming through the high windows, illuminating the workspace.

Outline

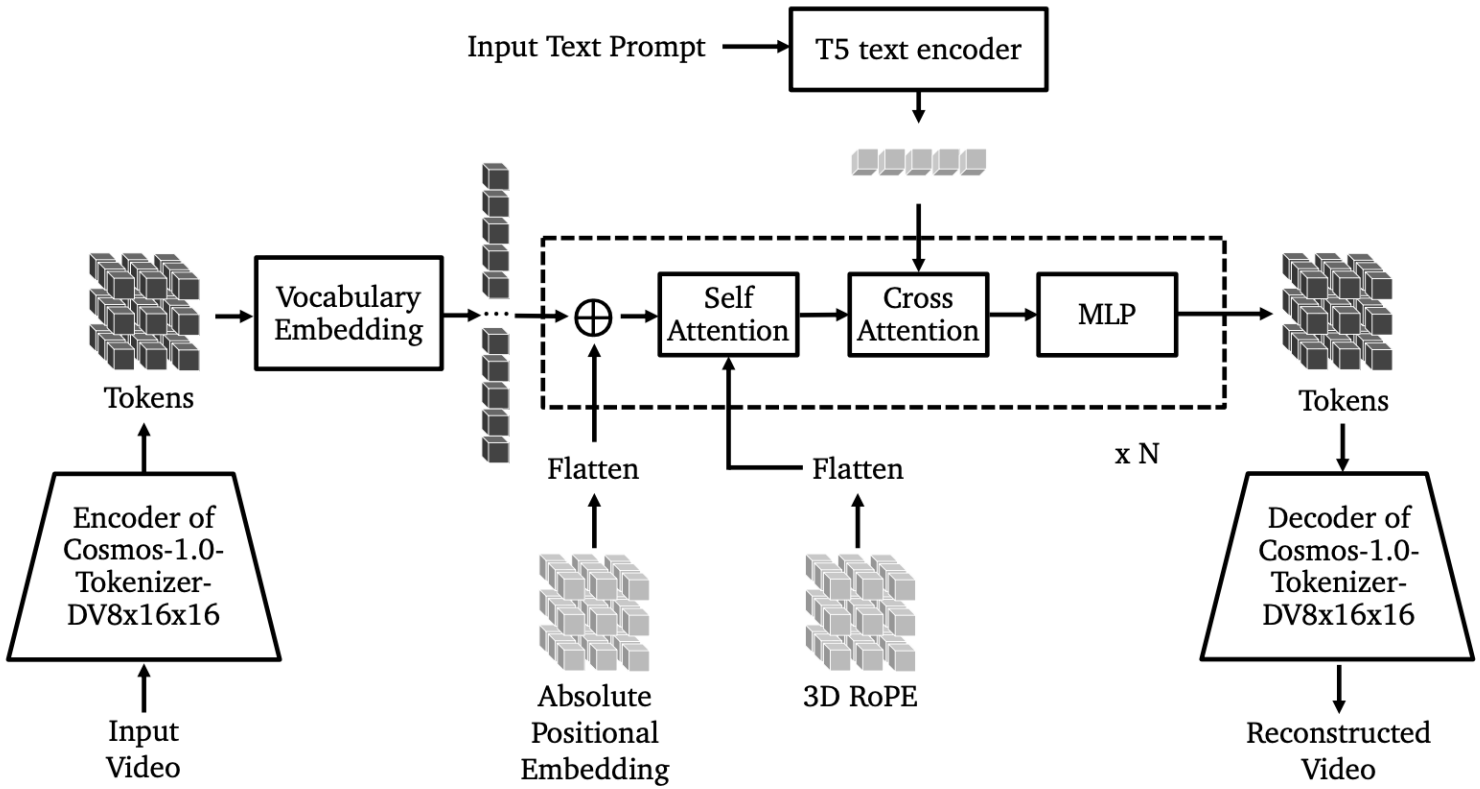
Formulation

World simulation generation as next-token prediction task (LLM style).

1. Tokenize video with cosmos tokenizer: $V = \{v_1, v_2, \ldots, v_n\}$
2. train to minimize:

$$\mathcal{L}_{NLL} = \sum_i -log\pi_\Theta(v_i|v_1, v_2, \ldots, v_{i-1})$$

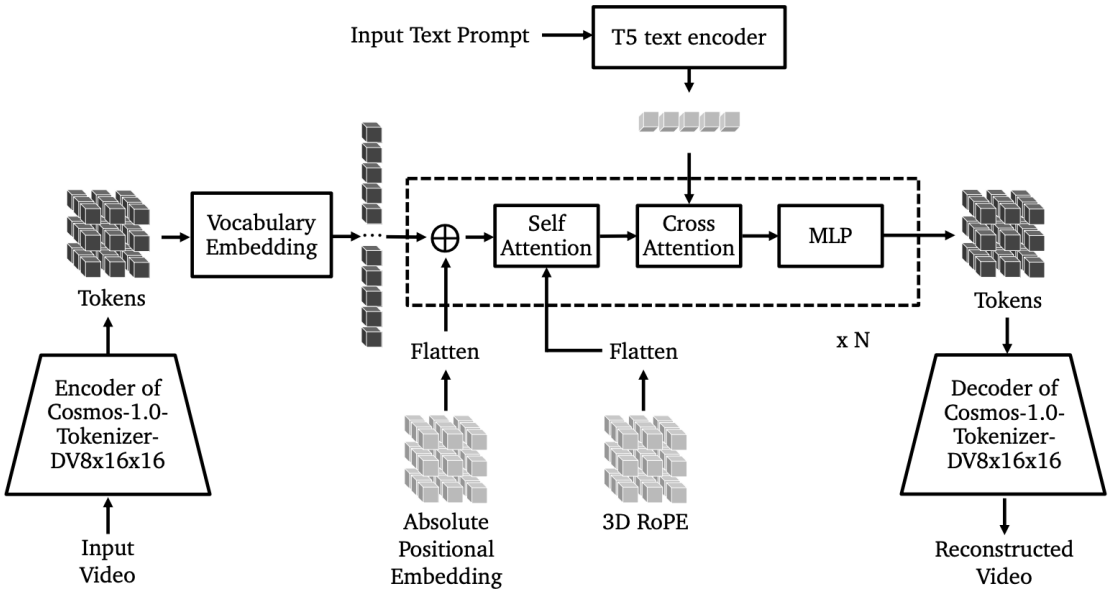
Architecture



3D positional embeddings: 3DRoPE and 3D Absolute Positional Embeddings (sin and cos across temporal, height, width)

Cross attention: after every self-attn

Architecture



Query-Key Normalization (QKNorm):

- Normalizes Q and K before computing dot product
- Dot product scaled by learnable param γ instead of $\sqrt{d_k}$

Z-loss: penalizes deviations of logits from 0

$$\mathcal{L}_{\text{z-loss}} = \lambda \sum_i z_i^2$$

Training Details

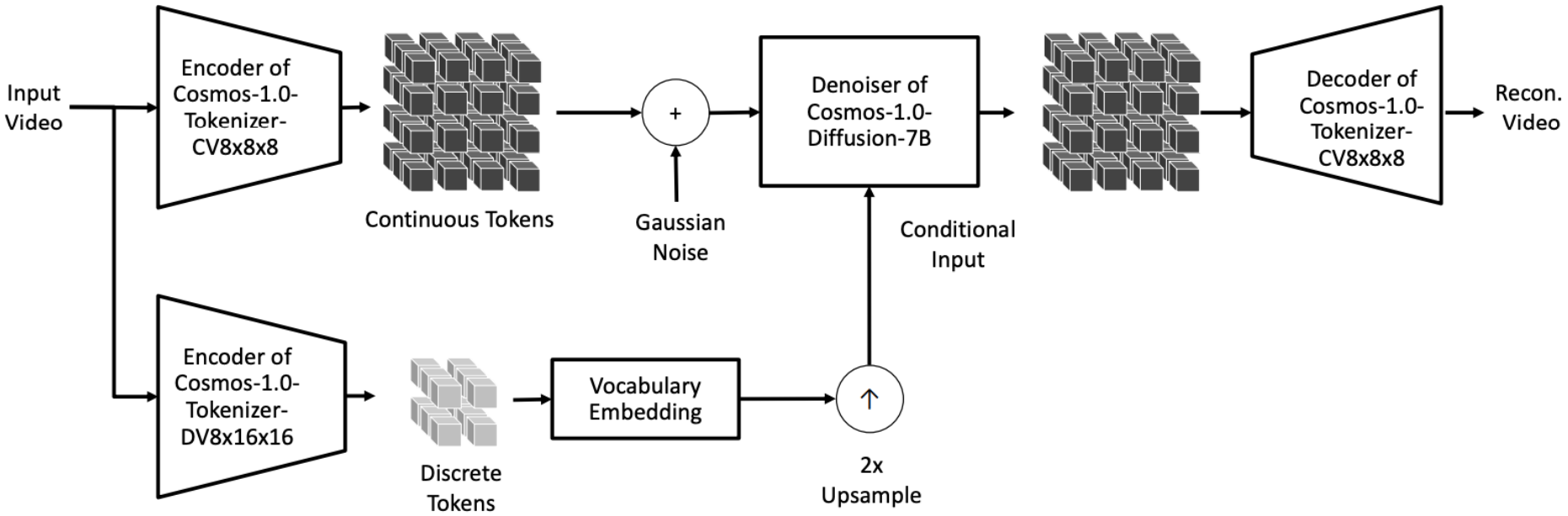
Pre-Training:

- **Stage 1:** Predict 16 future frames conditioned on first frame (context length $= 17$)
- **Stage 1.1:** Increase context length to 34 frames
- **Stage 2:** Text-conditioned generation with image-video batches similar to diffusion data

Cooling Down: Linearly decay learning rate to 0 while training on high-quality image-video pairs over 30k iterations.

Diffusion Decoder (Training)

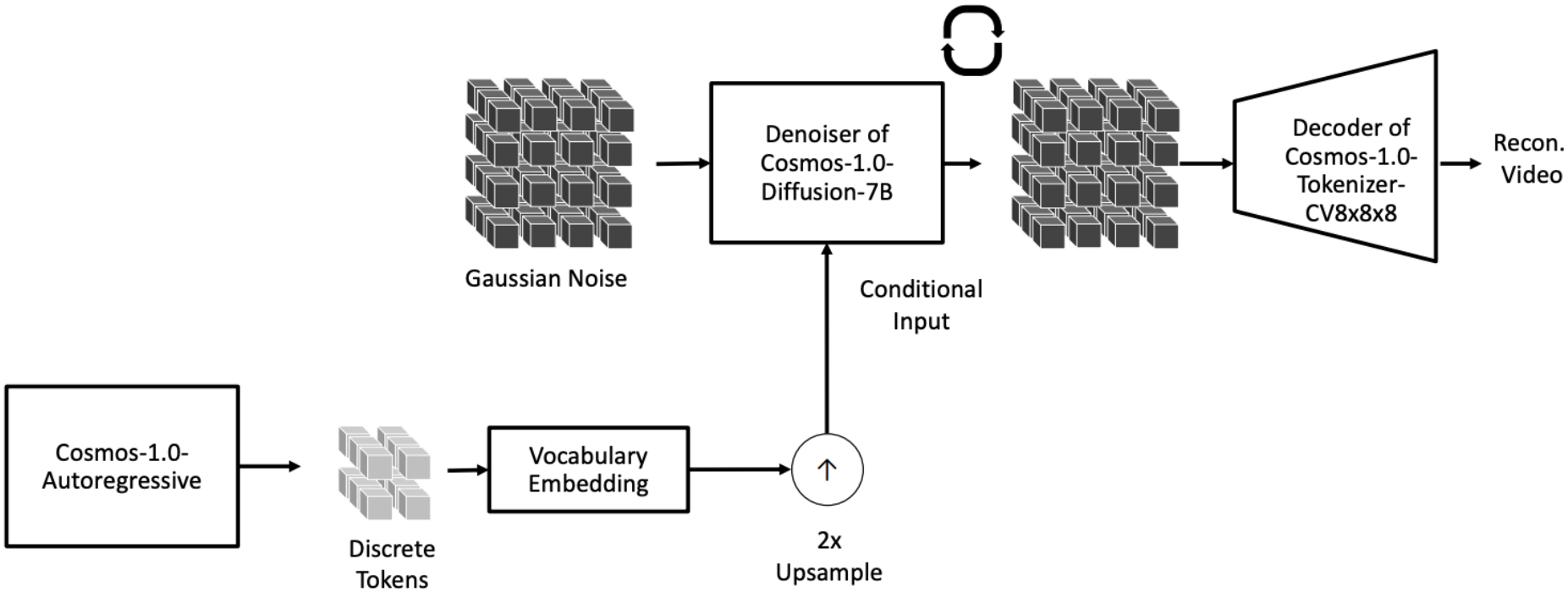
Problem: enc-dec tokenizer leads to blurriness, esp with discrete tokens
Solution: Diffusion decoder design.



Result: Higher-quality decoder that uses info in conditional input for denoising

Diffusion Decoder (Inference)

During inference, video tokens conditionally inputted to decoder.



AutoRegressive Model Results

5B




13B




Prompt: The video of a car moving forward, passing under a large overpass. The road is clear, and there are a few other cars visible in the distance. The weather appears to be sunny, and the time of day is daytime. The scene is set on a busy highway with concrete structures and greenery on the sides.

AutoRegressing and Duffusion Results

Method	Geometric Consistency		View Synthesis Consistency		
	Sampson error ↓	Pose estimation success rate (%) ↑	PSNR ↑	SSIM ↑	LPIPS ↓
VideoLDM (Blattmann et al., 2023)	0.841	4.4%	26.23	0.783	0.135
Cosmos-1.0-Diffusion-7B-Text2World	**0.355**	62.6%	**33.02**	**0.939**	**0.070**
Cosmos-1.0-Diffusion-7B-Video2World	0.473	**68.4%**	30.66	0.929	0.085
Cosmos-1.0-Autoregressive-4B	0.433	35.6%	32.56	0.933	0.090
Cosmos-1.0-Autoregressive-5B-Video2World	0.392	27.0%	32.18	0.931	0.090
Real Videos (Reference)	0.431	56.4%	35.38	0.962	0.054

Outline

Planner and Simulator Applications

Instruction-based video prediction:

Input: current video frame, text instruction

Output: video robot following instruction

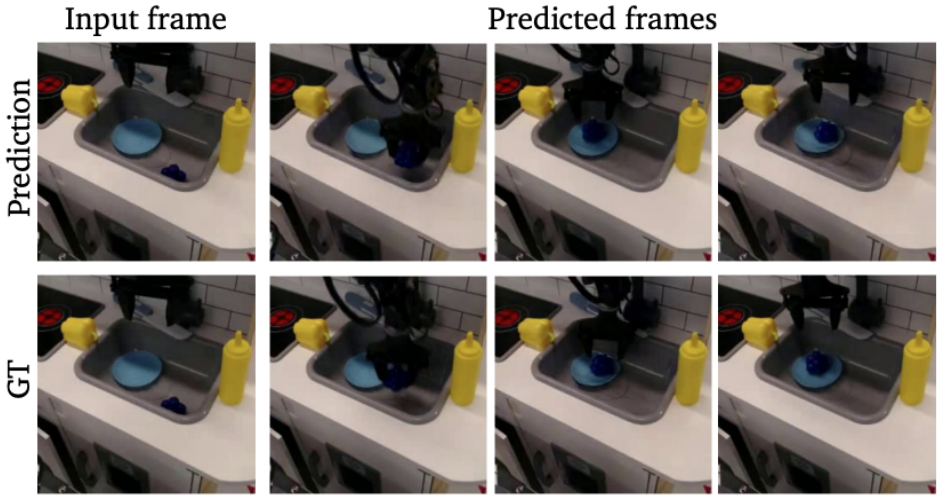
Prompt: Organize books by placing them vertically on a shelf.

Prompt: Fold a green fabric item on a table.

Action-based next-frame prediction:

Input: current video frame, action vector

Output: frame robot performing action



Datasets

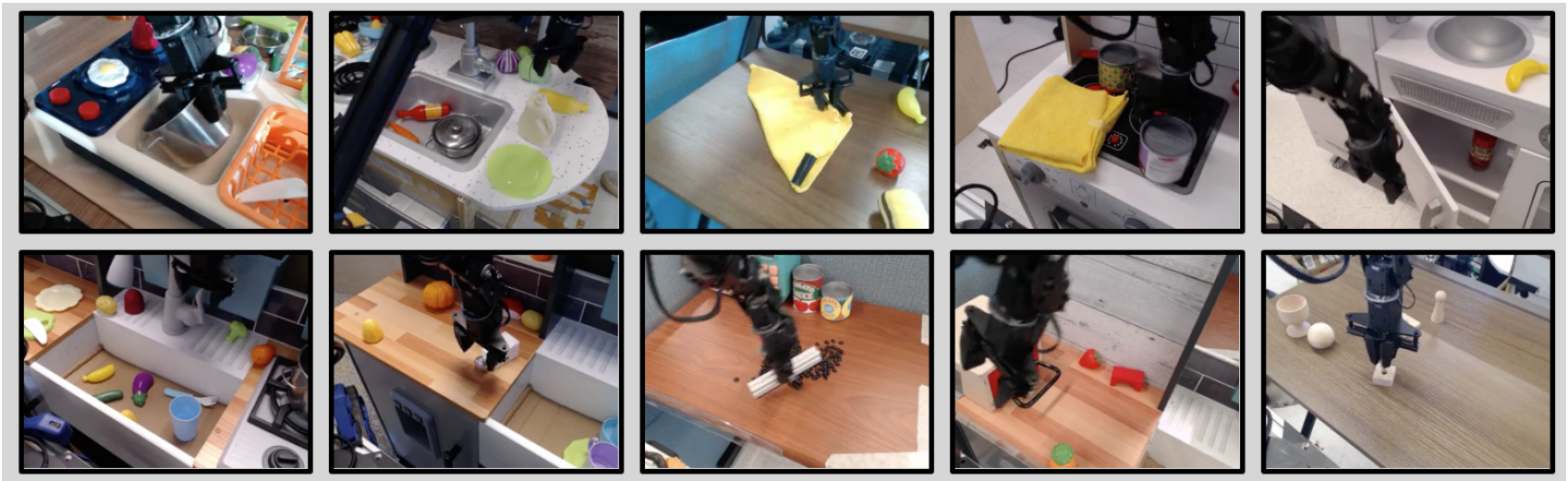
Instruction-based video prediction:

- Generate 200 hrs of EVE performing tasks
- Select 12k 1-9sec episodes
- Label actions, then upsample with VLM

Action-based next-frame prediction:

- Bridge dataset: 20k episodes at 5 FPS of 7-DOF arm performing tasks
- Each frame has action defined as 7-dim vector:

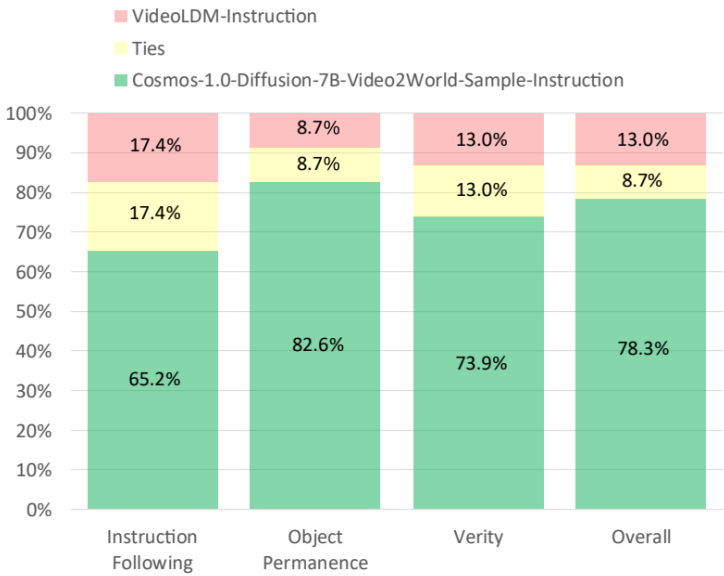
$$(\Delta x, \Delta y, \Delta z, \Delta\theta_r, \Delta\theta_p, \Delta\theta_y, \Delta_{\text{Gripper}})$$



Fine-tuning and Evaluation

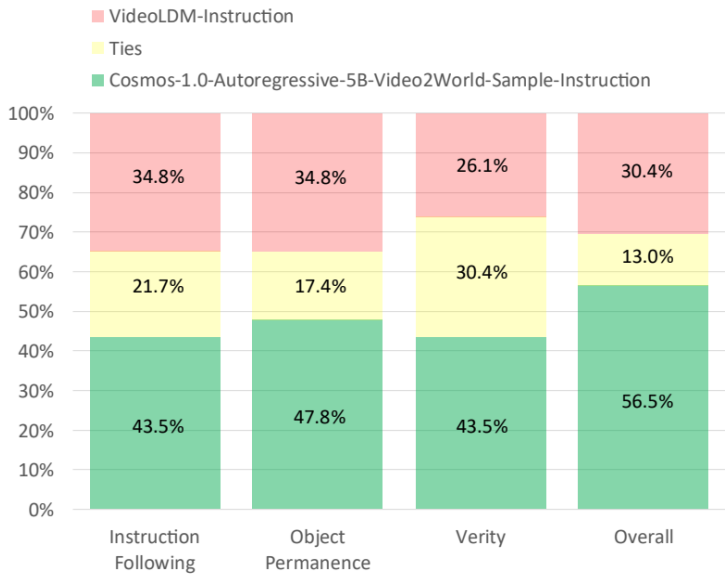
Instruction-based video prediction:

- Compute T5 embeddings for instructions
- *added via cross-attn*



Action-based next-frame prediction:

- Add MLP to project action-vector to tensor
- *added via cross-attn*



Method	PSNR ↑	SSIM ↑	Latent L2 ↓	FVD ↓
IRASim-Action	19.13	0.64	0.38	593
Cosmos-1.0-Autoregressive-5B-Video2World-Sample-ActionCond	19.95	0.80	0.36	434
Cosmos-1.0-Diffusion-7B-Video2World-Sample-ActionCond	**21.14**	**0.82**	**0.32**	**190**

Thank you!

Have a great rest of your Day!!!