Yan: Foundational Interactive Video Generation

Yan Team – Tencent
Tech Report

David He, Jinhoo Kim, Wonho Choi

Georgia Tech

Outline

- Problem Statement & Background
- Related Works
- Data
- Approach & Experiments: Yan-Sim, Yan-Gen, Yan-Edit
- Strengths
- Limitations

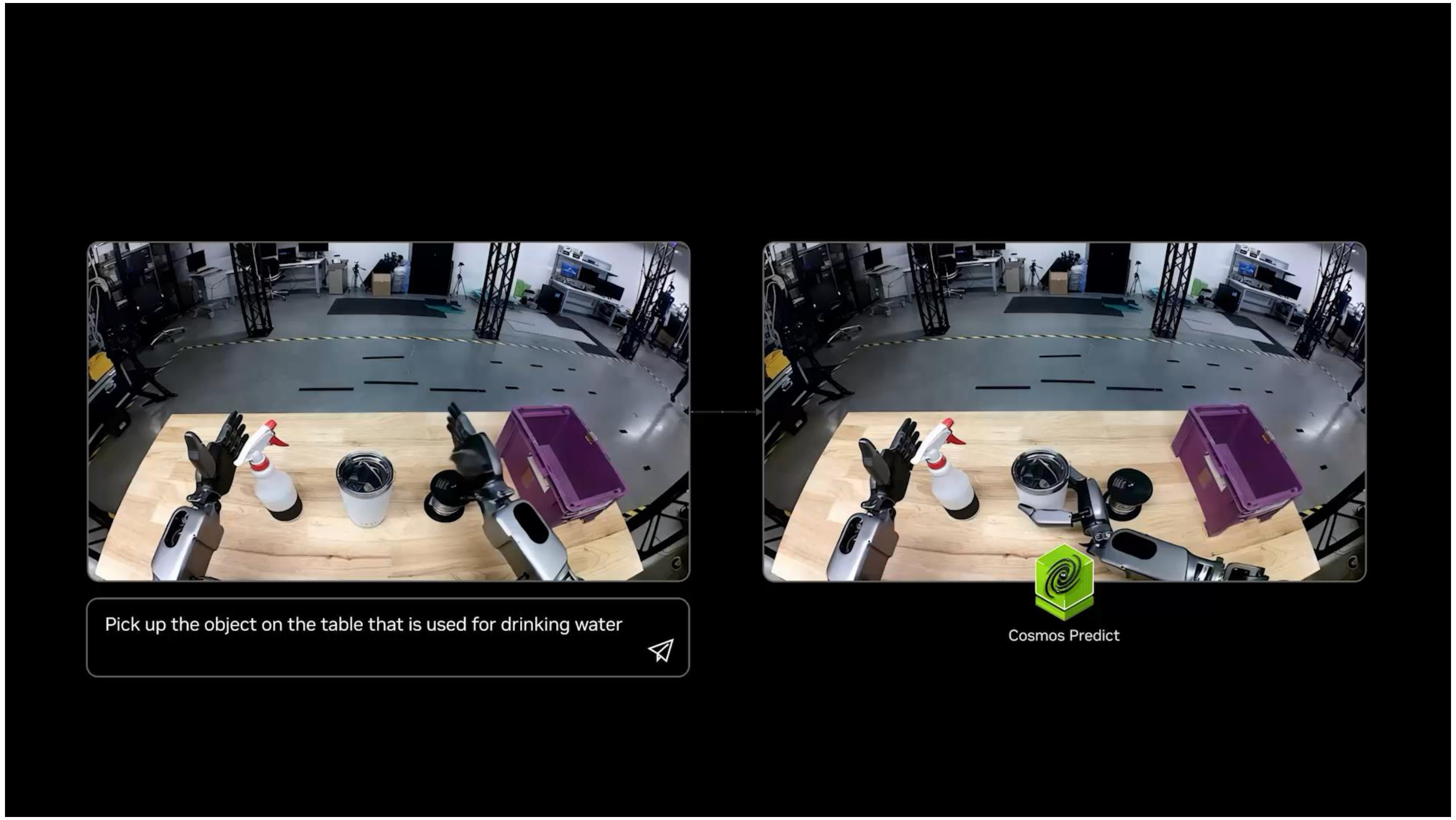
Interactive Video Generation

Problem: Generate (next observation) conditioned on (previous observations, previous actions, current action)

Genie 3: POV action camera of a tan house being painted by a first-person agent with a paint roller

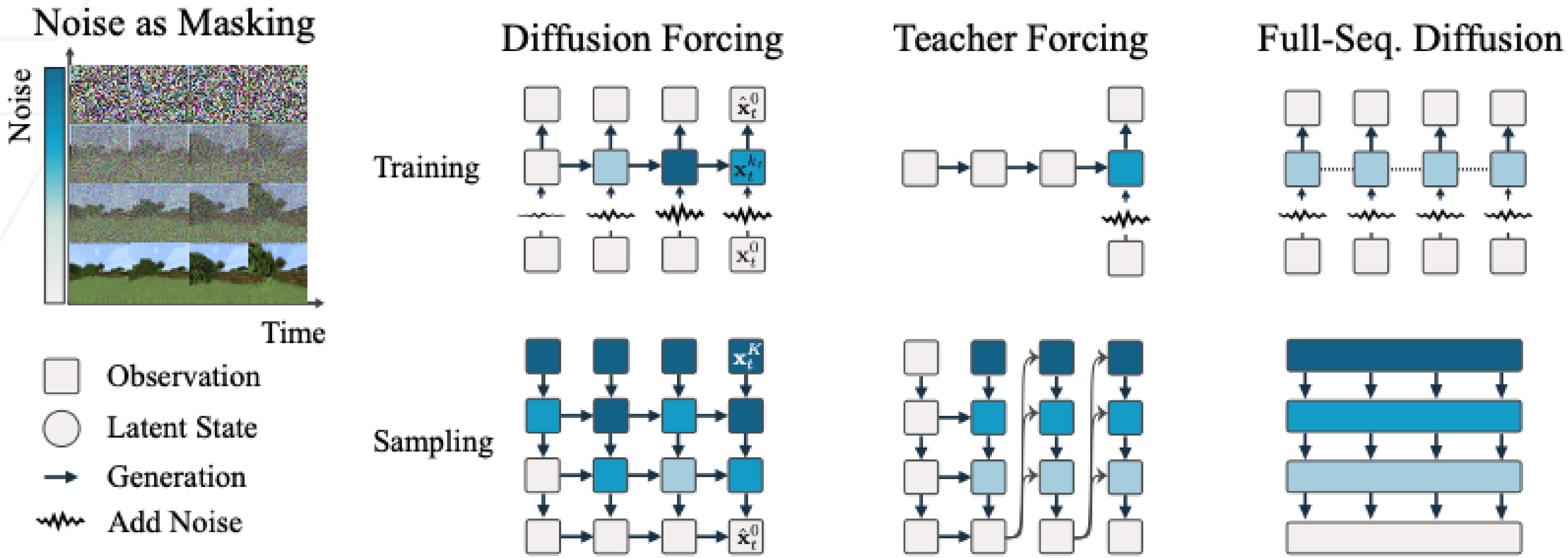
Interactive Video Generation

Motivation: Generating interactive world simulators to train embodied agents.



Background: Diffusion Forcing

Denoise the next token with *independent* per-token noise levels on previous tokens



Background: Diffusion Forcing

Algorithm 1 Diffusion Forcing Training

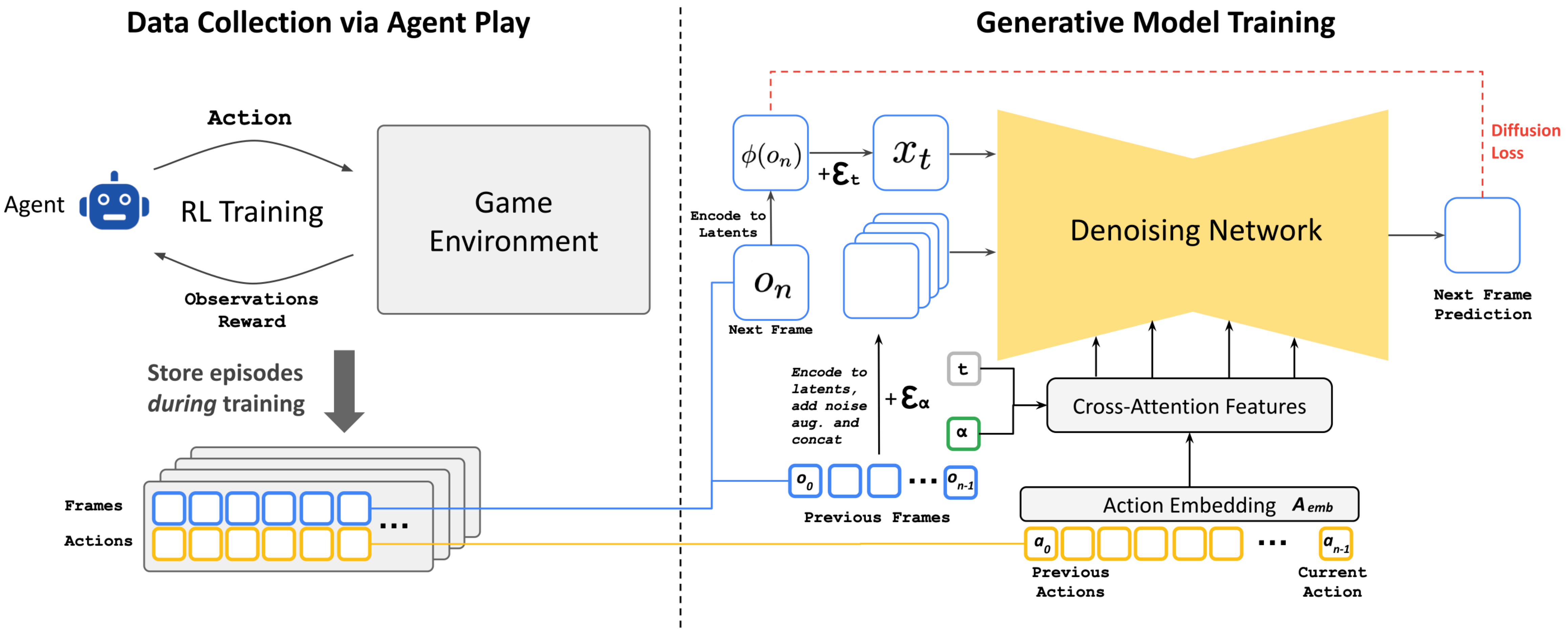
loop
  Sample tajectory of observations $(\mathbf{x}_1, ..., \mathbf{x}_T)$.
  for $t = 1, ..., T$ **do**
    Sample independent noise level $k_t \in \{0, 1, ..., K\}$
    $\mathbf{x}_t^{k_t} = \text{ForwardDiffuse}(\mathbf{x}_t, k_t)$
    Define $\epsilon_t = \frac{\mathbf{x}_t^{k_t} - \sqrt{\bar{\alpha}_{k_t}}\mathbf{x}_t}{\sqrt{1-\bar{\alpha}_{k_t}}}$
    Update $\mathbf{z}_t \sim p_\theta(\mathbf{z}_t | \mathbf{z}_{t-1}, \mathbf{x}_t^{k_t}, k_t)$.
    Set $\hat{\epsilon}_t = \epsilon_\theta(\mathbf{z}_{t-1}, \mathbf{x}_t^{k_t}, k_t)$
  end for
  $L = \text{MSELoss}([\hat{\epsilon}_1, ..., \hat{\epsilon}_n], [\epsilon_1, ..., \epsilon_n])$
  Backprop with L and update θ
end loop

Algorithm 2 DF Sampling with Guidance

Input: Model θ, scheduling matrix $\mathcal{K}$, initial latent $\mathbf{z}_0$, guidance cost $c(\cdot)$.
Initialize $\mathbf{x}_1, \dots, \mathbf{x}_T \sim \mathcal{N}(0, \sigma_K^2 I)$.
for row $m = M - 1, ..., 0$ **do**
  for $t = 1, \dots, T$ **do**
    $\mathbf{z}_t^{\text{new}} \sim p_\theta(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{x}_t, \mathcal{K}_{m+1,t})$.
    $k \leftarrow \mathcal{K}_{m,t}$, $\mathbf{w} \sim \mathcal{N}(0, \mathbf{I})$.
    $\mathbf{x}_t^{\text{new}} \leftarrow \frac{1}{\sqrt{\alpha_k}}(\mathbf{x}_t - \frac{1-\alpha_k}{\sqrt{1-\bar{\alpha}_k}}\epsilon_\theta(\mathbf{z}_t^{\text{new}}, \mathbf{x}_t, k)) + \sigma_k \mathbf{w}$
    Update $\mathbf{z}_t \leftarrow \mathbf{z}_t^{\text{new}}$.
  end for
  $\mathbf{x}_{1:H} \leftarrow \text{AddGuidance}(\mathbf{x}_{1:H}^{\text{new}}, \nabla_\mathbf{x} \log c(\mathbf{x}_{1:H}^{\text{new}}))$
end for
Return $\mathbf{x}_{1:T}$.

GameNGen

$$p(o_n | o_{<n}, a_{<n})$$



PlayGen

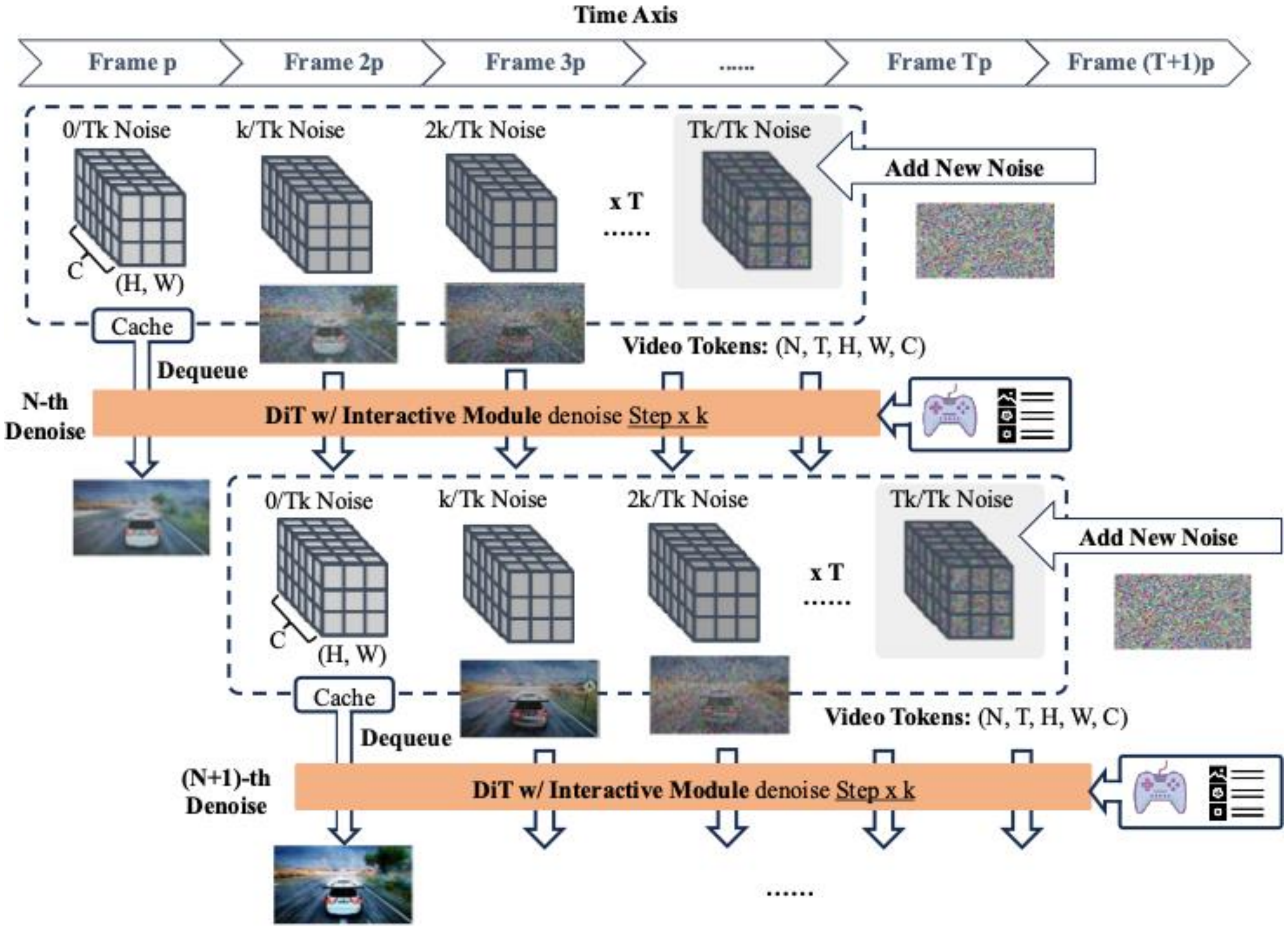
Algorithm 2 Balanced Data Sampling

Input: Collected transition dataset $\mathcal{D}$, number of clusters $k \in \mathbb{N}$.
Output: Balanced transition dataset $\mathcal{D}_{\text{balanced}}$.
Calculate the transition characteristics (e.g., position distribution) based on e_t as a feature vector for each sample in $\mathcal{D}$.
Cluster all samples into k clusters based on the feature vectors, and obtain k cluster centers $\{\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_k\}$.
Formulate a linear equation:

$$b_1\mathbf{c}_1 + b_2\mathbf{c}_2 + \cdots + b_k\mathbf{c}_k = \mathbf{y},$$

where $\mathbf{y}$ is the target balanced transition characteristics, i.e., balanced transition characteristics (e.g., balanced position distribution).
Solve the linear equation using the non-negative least squares method to obtain an approximate non-negative integer solution $\{b_1, b_2, \cdots, b_k\}$, $b_i \in \mathbb{N}$.
for $i = 1, 2, \cdots, k$ **do**
    Sample b_i samples from the i^{th} cluster.
end for
Obtain $\mathcal{D}_{\text{balanced}}$ consisting of $\sum_{i=1}^{k} b_i$ samples.

The Matrix



Time Axis
Frame p
Frame 2p
Frame 3p
......
Frame Tp
Frame (T+1)p
0/Tk Noise
k/Tk Noise
2k/Tk Noise
Tk/Tk Noise
x T
Add New Noise
C
(H, W)
Cache
Dequeue
Video Tokens: (N, T, H, W, C)
N-th Denoise
DiT w/ Interactive Module denoise Step x k
0/Tk Noise
k/Tk Noise
2k/Tk Noise
Tk/Tk Noise
x T
Add New Noise
C
(H, W)
Cache
Dequeue
Video Tokens: (N, T, H, W, C)
(N+1)-th Denoise
DiT w/ Interactive Module denoise Step x k
......

Related Works

Paper	Date (arxiv)	Game	Novelty	Limitations
GameNGen	08/2024	Doom	First “high-quality” generative game engine	Limited context window, data collection is sensitive to reward function
PlayGen	12/2024	Super Mario Bros, Doom	Data Balancing	RNN also struggles with long context
The Matrix	12/2024	CyberPunk, Forza	Sliding Window Denoising	Human data collection is not scalable
Yan	08/2025	Yuan Meng Star	Yan-Edit	Only trained on Yuan Meng Star

A Note on Evaluation

Yan has no quantitative evaluations!

How have other works evaluated?

Image Quality: LPIPS, PNSR, FID

Video Quality: FVD

Human Evaluation: choose real game when given a real and generated video clip

Playability: train a separate model to predict $p(a_n|o_n, o_{n+1})$, and then use this to measure the compatibility of frames and actions

Data (Engineering)

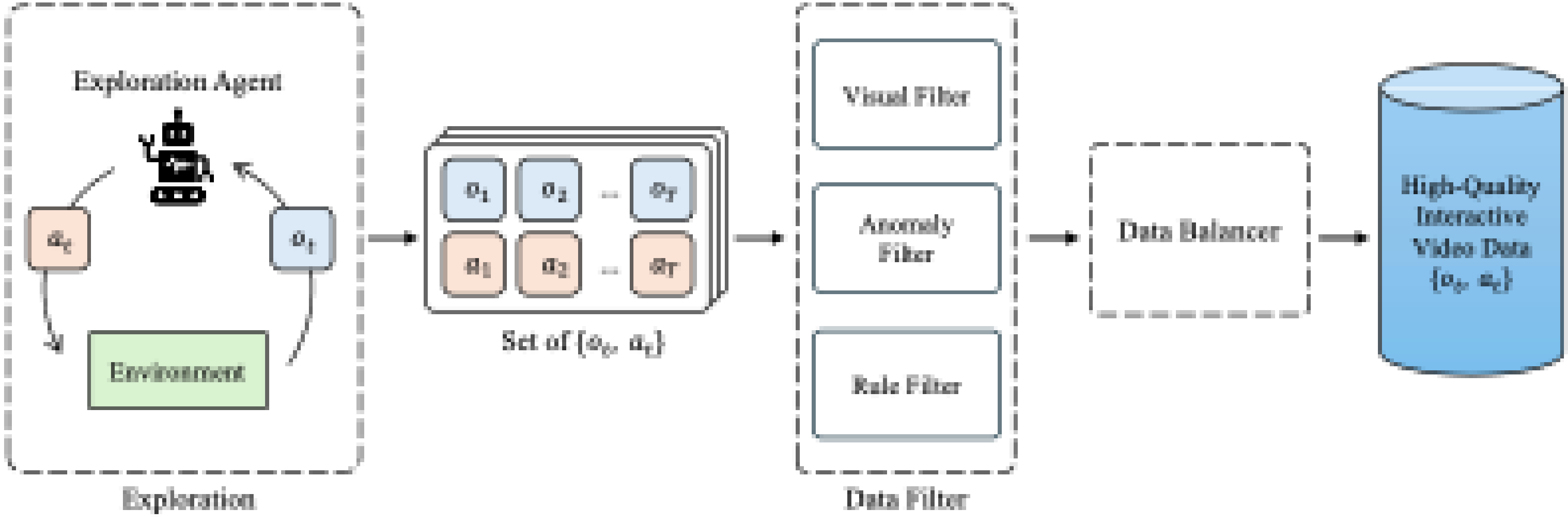
	Resolution	FPS	Frame-wise	Action Space	Scale
The Matrix (Feng et al., 2024)	720P	60	✓	5	792M
PlayGen (Yang et al., 2024)	128P	30	✓	5	250M
GameGenX (Che et al., 2024)	720P-4K	1-24	×	×	192M
GameFactory (Yu et al., 2025b)	360P	16	✓	9	4M
Matrix-Game (Zhang et al., 2025)	720P	16	✓	7	50M
Ours	1080P	30	✓	8	400M

Collection Strategy for Diversity: random + PPO agents for breadth and depth

Data Filters: remove occlusions, engine lag, game loading frames, etc..

Data Balancing: stratified sampling across xyz coordinate, is the agent alive?, etc…

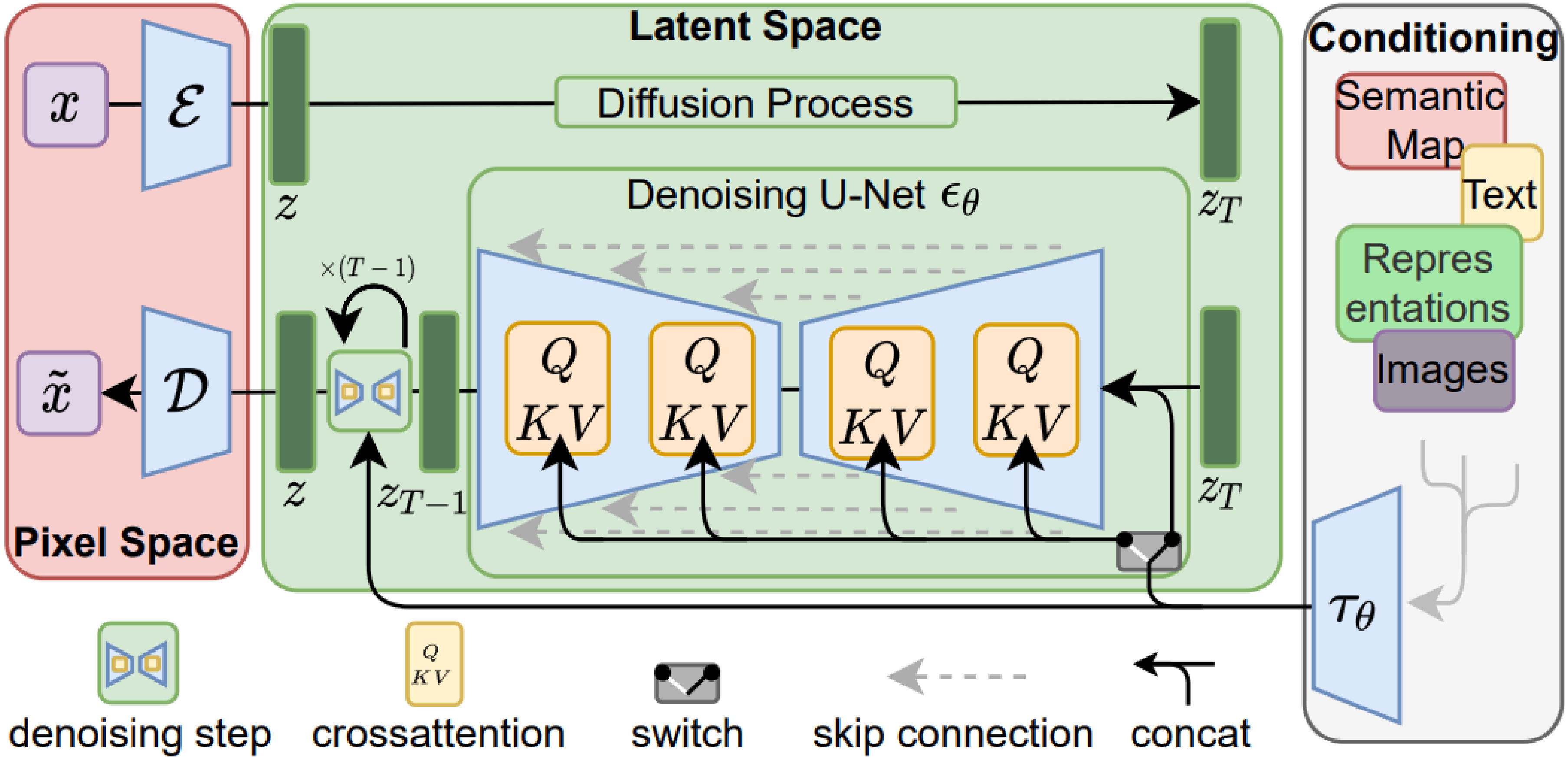
1080p screenshots at 30 FPS, 400 million frames in 90 scenarios/styles



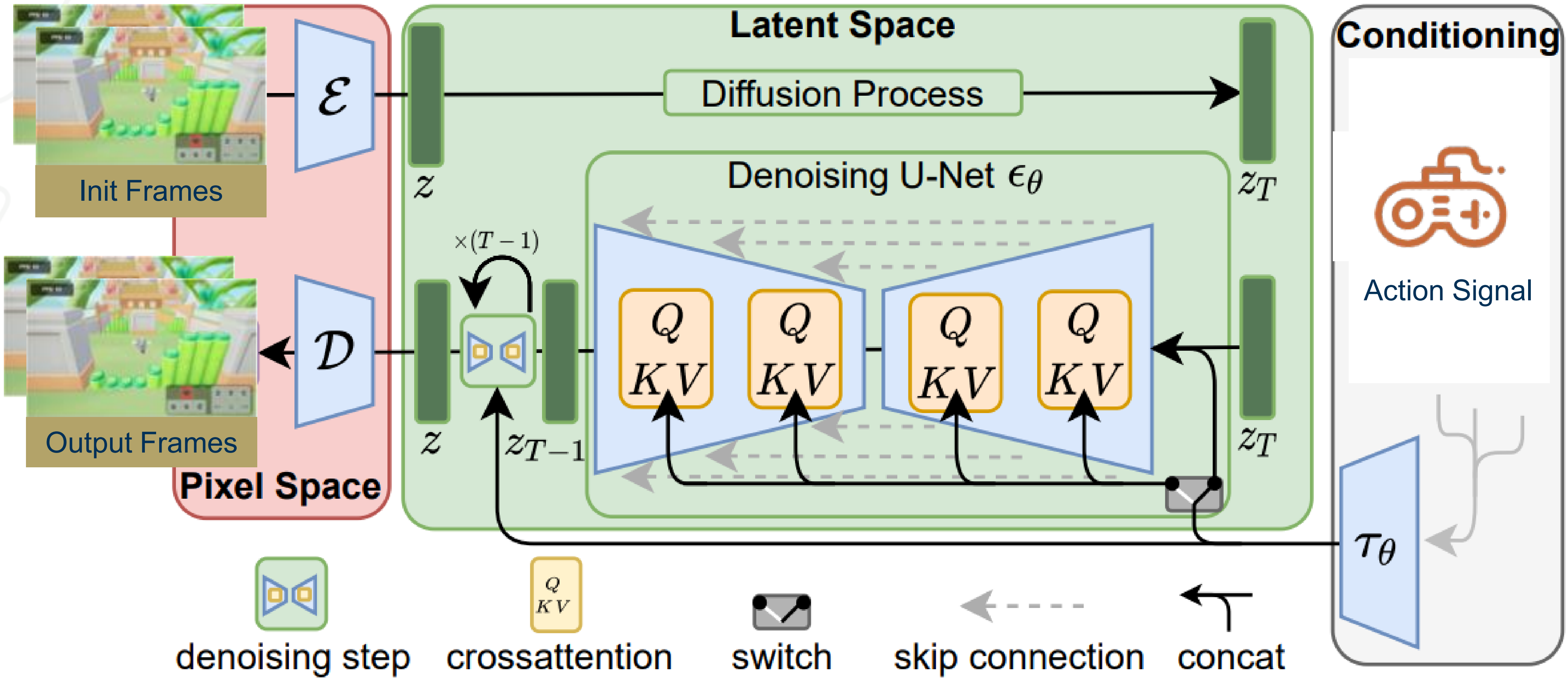
Yan-Sim: AAA-level Simulation

- Real-time high-resolution world simulator
- Stable Diffusion Architecture

Stable Diffusion



Stable Diffusion -> Yan-Sim



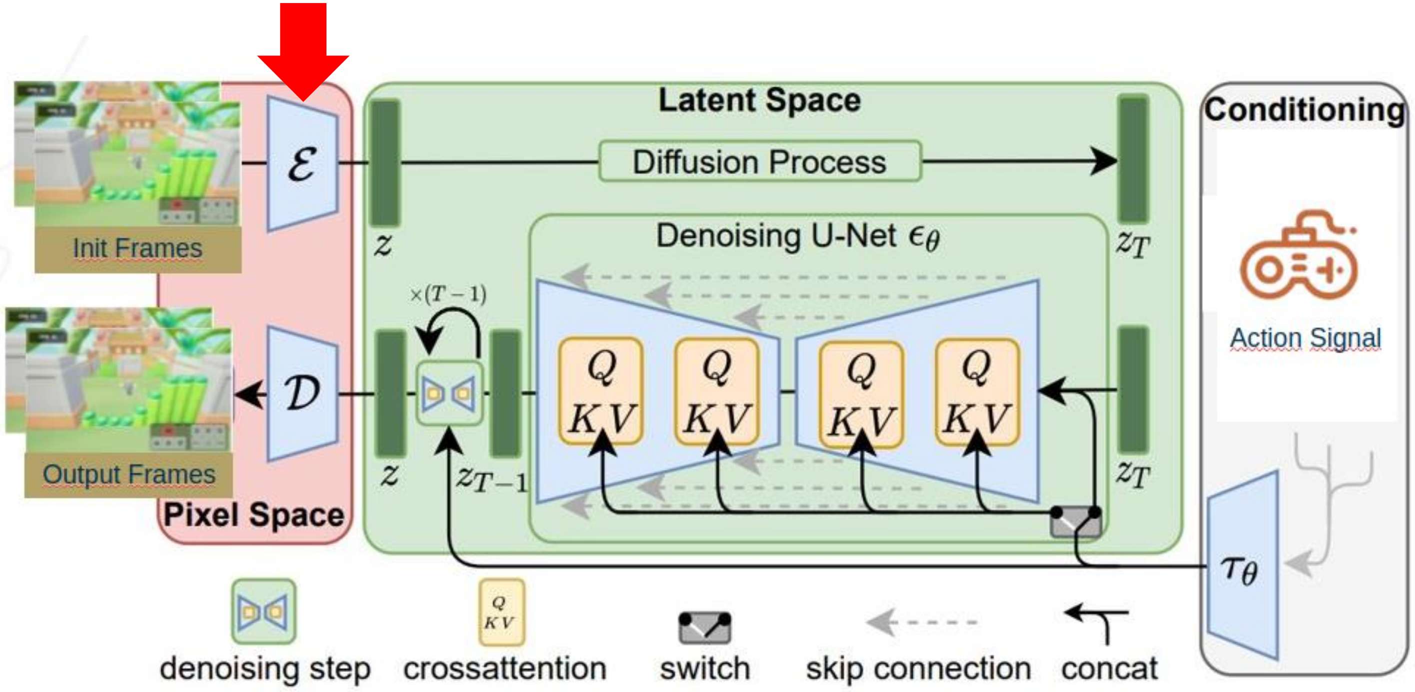
Yan-Sim: Improvement from Stable Diffusion

- Spatial-temporal consistency
  - Enhanced spatial-temporal compression
  - Spatial, action cross, and temporal attention blocks
- Faster Inference
  - Faster Decoder
  - Deterministic Sampler and less denoising steps
  - Shift Window Denoising Inference
- Other Optimizations
  - Pruning and Quantization

Yan-Sim: Spatial-temporal consistency

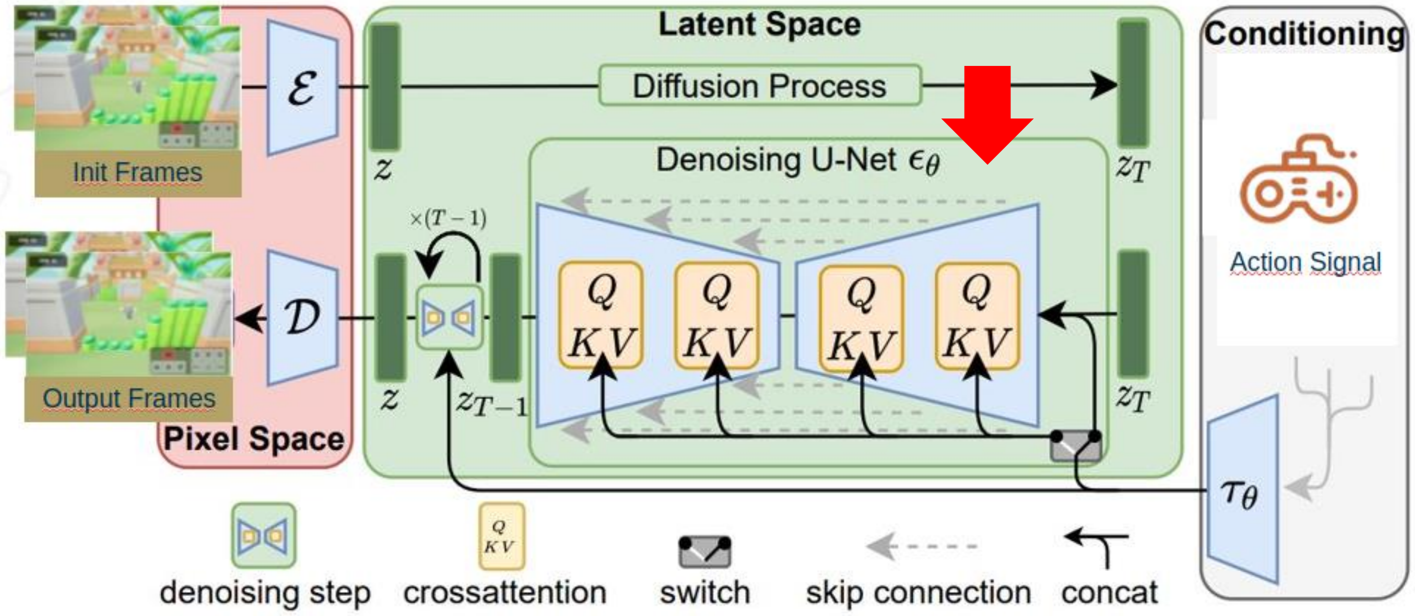
- Modification on VAE
  - Increased spatial downsampling rate from 8 to 32
  - Increased temporal downsampling rate from 1 to 2
  - 1 x 8 x 8 -> 2 x 32 x 32
  - Increased latent channel dimension from 4 to 16

-> Higher information density with better spatio-temporal compression



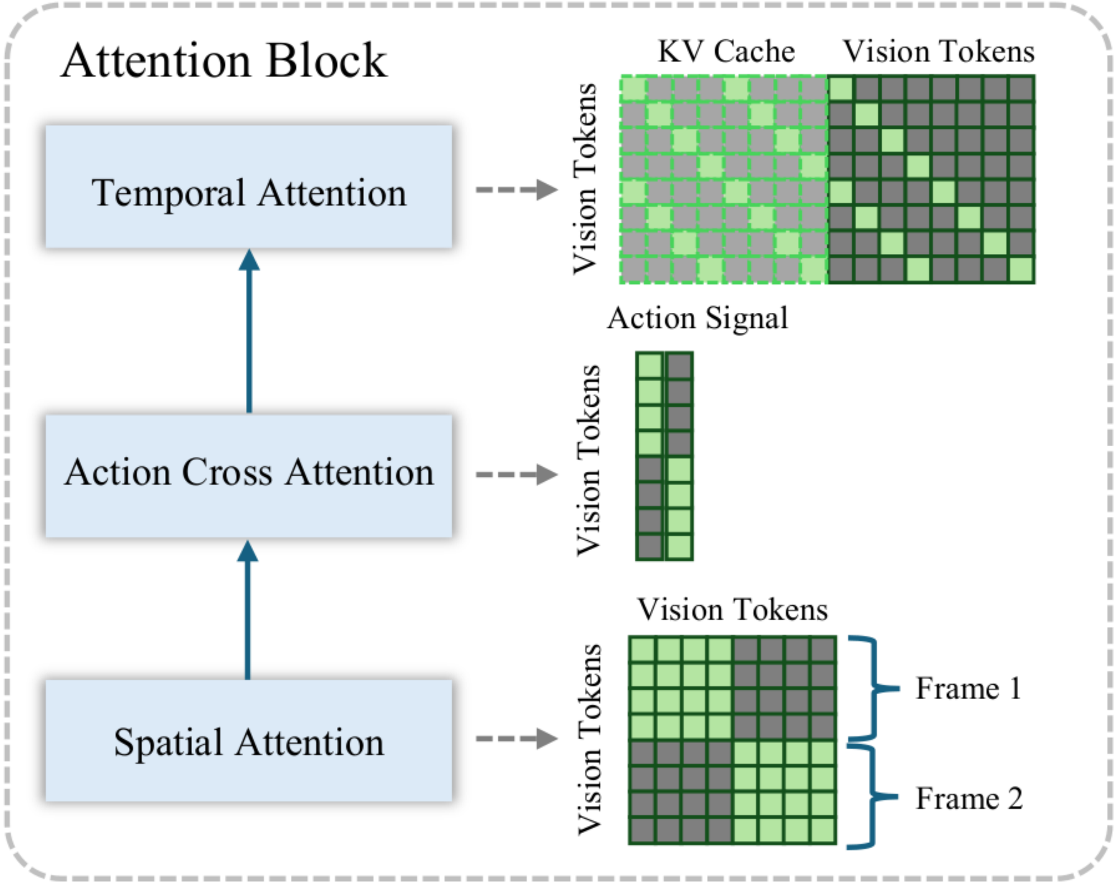
Yan-Sim: Spatial-temporal consistency

- Spatial, Action Cross, and Temporal Attention for each block of Denoising U-Net



Yan-Sim: Spatial-temporal consistency

- Spatial Attention
  - Spatial positions within the same frame
  - Same as SD
- Action Cross Attention
  - Action-conditioned cross attention
  - Same as the text cross attention from SD
  - Action Signals processed by a MLP layer to generate an action token (Dim=768)
- Temporal Attention
  - 1D temporal attention
  - Inter-frame dependencies
  - Causal (Not bi-directional)

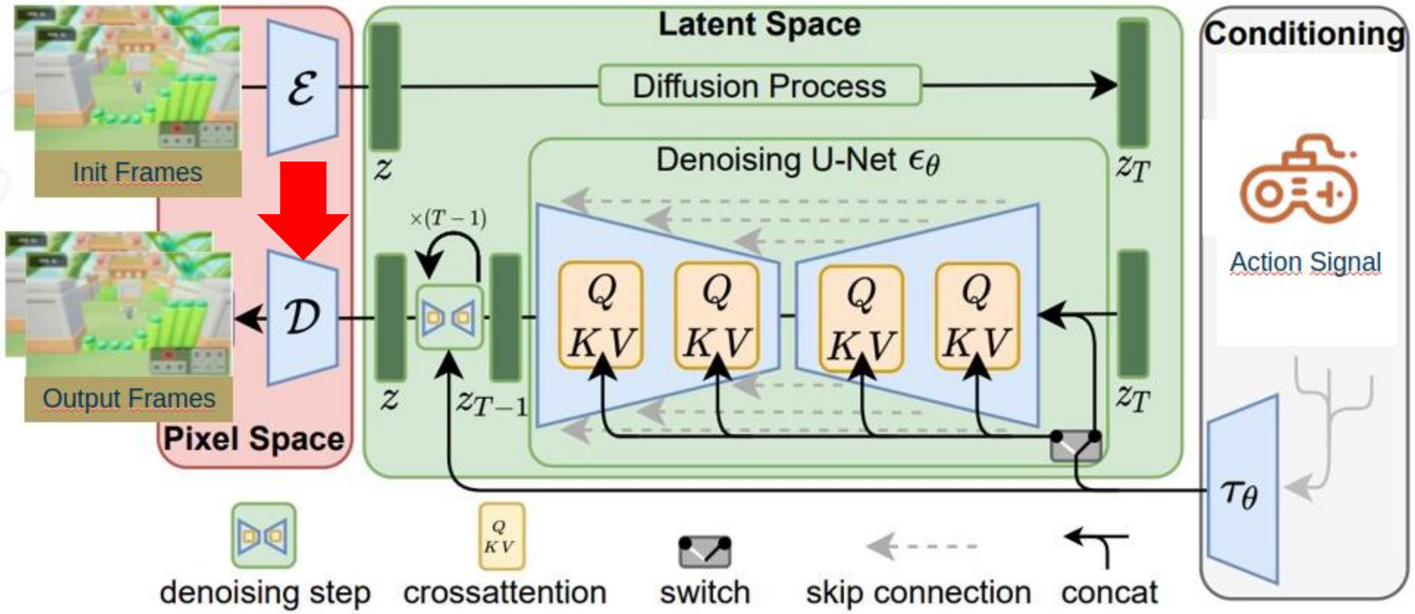


Yan-Sim: Faster Inference

- Faster Inference

  = Lightweight Decoder

- Modification on Decoder
  - Reducing a layer per up block
  - Added a single layer up block and a pixel shuffle layer at the end for alignment

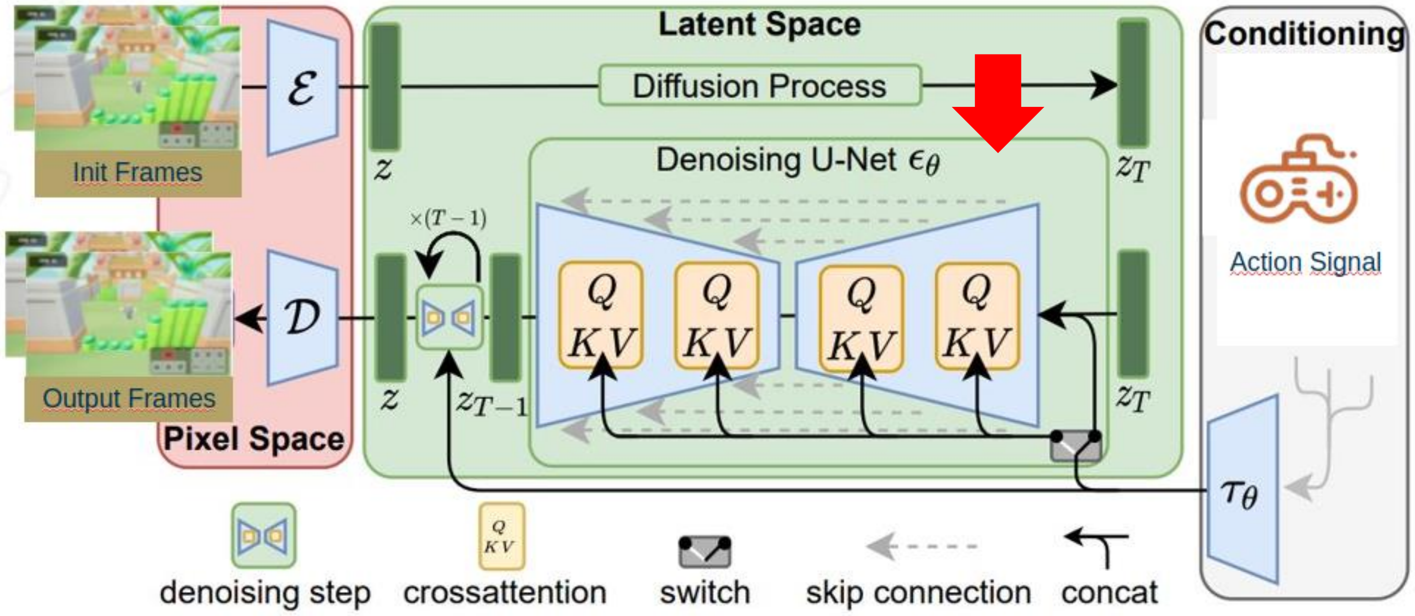


Georgia Tech

Yan-Sim: Faster Inference

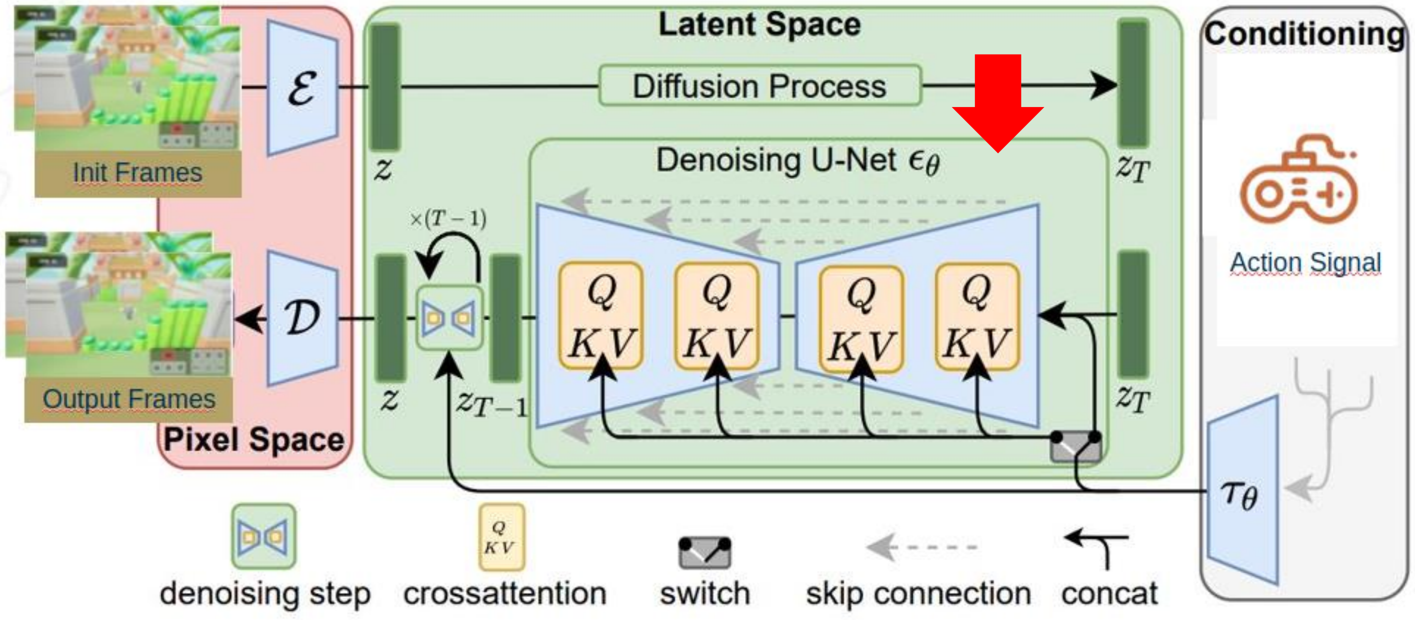
- Denoising Diffusion Implicit Models (DDIM)
  - Deterministic sampler (first-order ODE instead of linear multistep ODE)
  - Requires fewer steps

- 4 Denoising Steps
  - Standard SD used 50 denoising steps



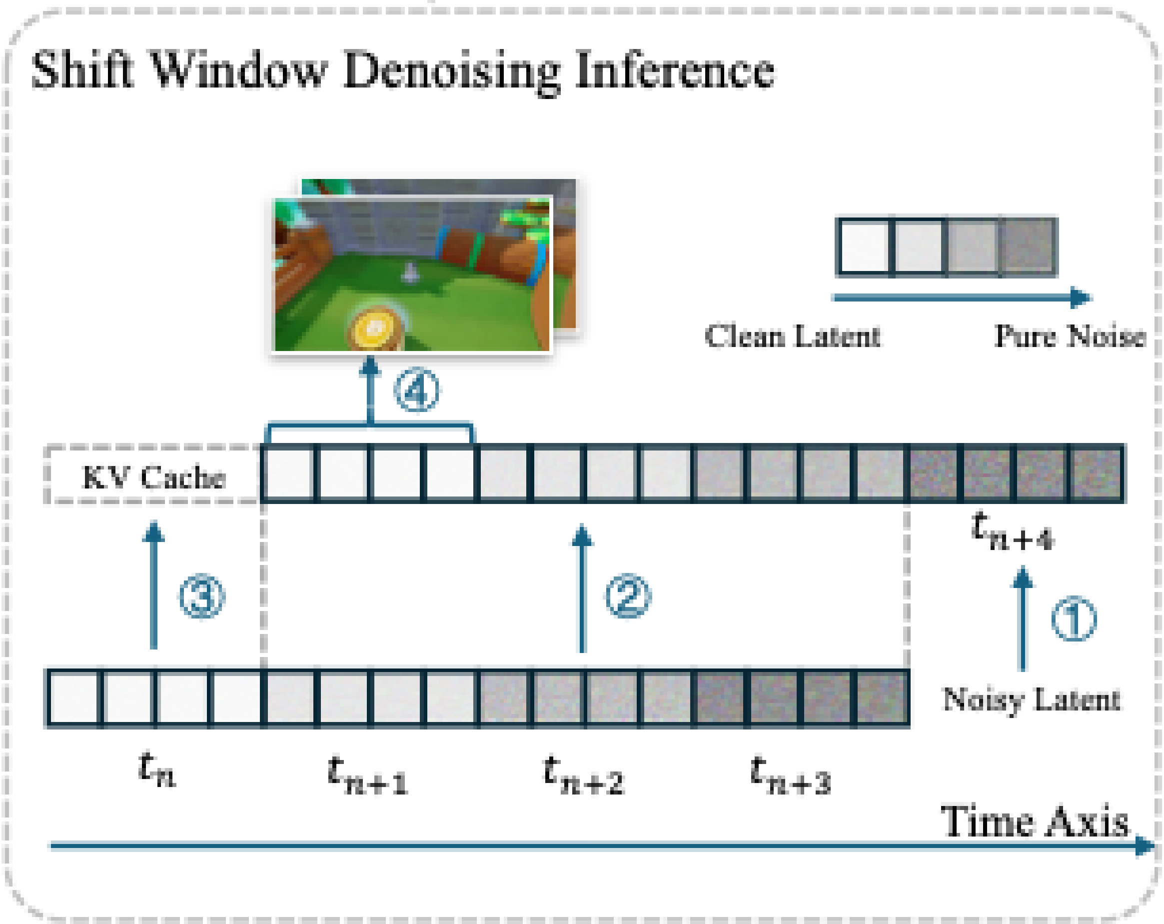
Yan-Sim: Faster Inference

- Shift Window Denoising Inference



Yan-Sim: Faster Inference

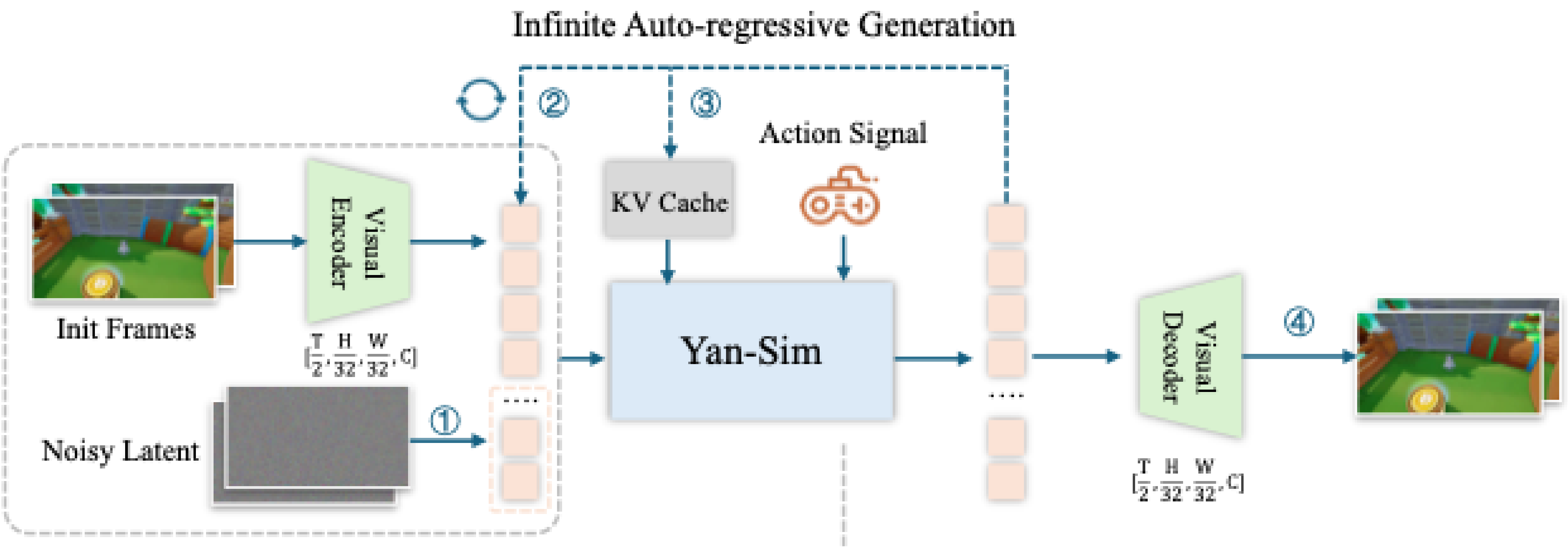
- Shift Window Denoising Inference
- Inference step processes a window of frames
- KV Cache is used to store previous attention states



Yan-Sim: Pruning and Quantization

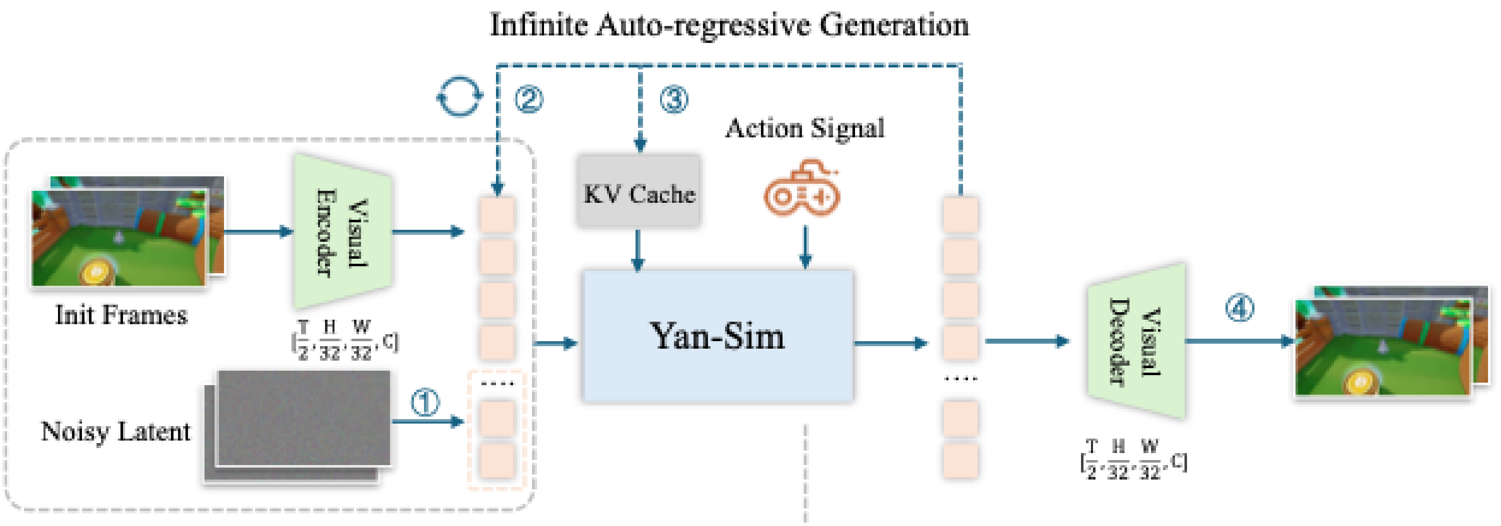
- Structural pruning to UNet
- FP8 quantization of GEMMs (1.5-2x speed up)
- Cuda graph for kernel-launch overhead elimination and triton-based custom Kernels (1.15x speed up)
- Running two models on sperate GPUs to avoid serial inference

Yan-Sim: Architecture



Yan-Sim: Training

- VAE Training
- Diffusion Model Training



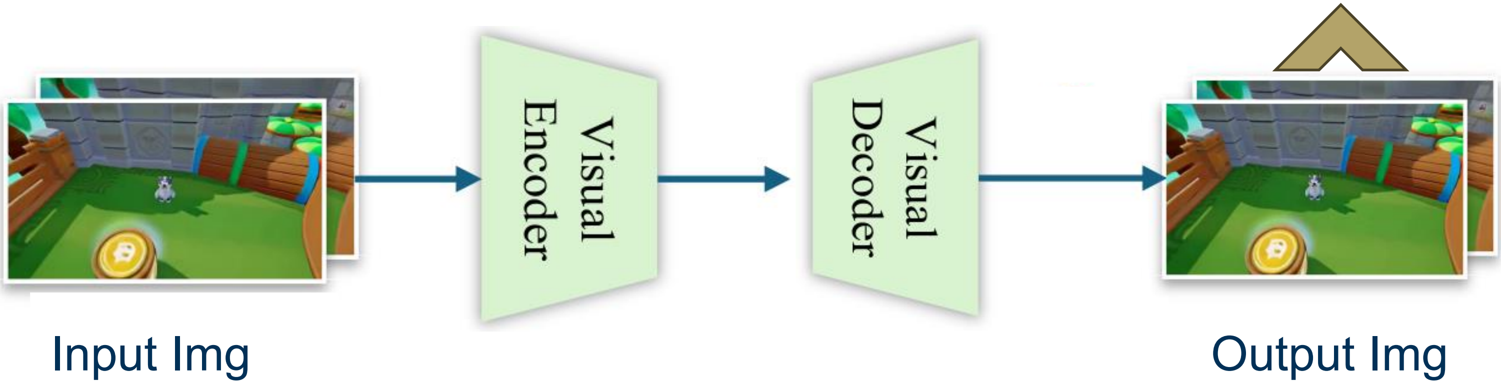
Yan-Sim: Training - 1) VAE Training

- VAE Training
  - Mean Squared Error (MSE)

= ||  -  ||

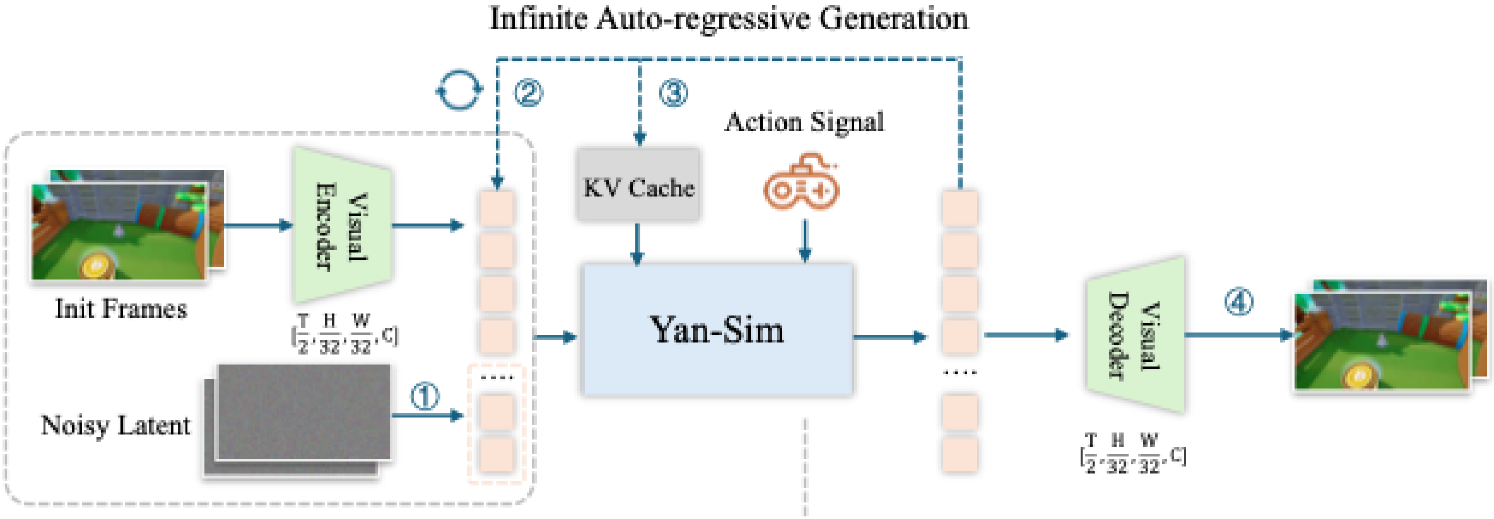
  - Learned Perceptual Image Path Similarity (LPIPS)
    - Feature distance

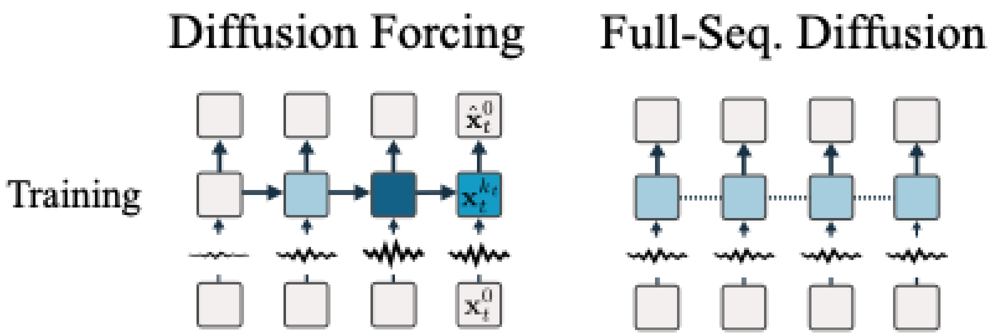
= ||f()-f()||



Yan-Sim: Training – 2) Diffusion Model Training

- Diffusion Model Training
  - Follows SD framework
  - Diffusion Forcing strategy
    - Denoise a set of tokens with independent per-token noise levels





Yan-Sim: Results

- Visual Quality
- Motion Consistency
- Accurate Mechanism Simulation
- Long Video Generation Capability

Comparison to other simulation framework

Video Length: Autoregressive generation ≠ “infinite” video length
Resolution: hyperparameter for the data and VAE decoder
Real Time: questionable, given that training data is 30 FPS
Low Latency: pruning + quantization + torch.compile

	Video Length	Resolution	Real Time	Low Latency
The Matrix (Feng et al., 2024)	Infinite	720p	✓ (16fps)	×
PlayGen (Yang et al., 2024)	Infinite	128p	✓ (20fps)	✓ (0.05s)
Genie 2 (Parker-Holder et al., 2024)	10 - 20s	360p	×	×
GameFactory (Yu et al., 2025b)	Infinite	640p	×	×
Matrix-Game (Zhang et al., 2025)	Infinite	720p	×	×
Genie 3 (Ball et al., 2025)	few minutes	720p	✓ (24fps)	✓
Yan-Sim	Infinite	**1080p**	✓ (**60fps**)	✓ (0.11s)

Yan-Gen: Multi-Modal World Generation Model

- Real-time and interactive world generator
- Adaptive synthesis across varied scenarios

Wan: Open and Advanced Large-Scale Video Generative Models

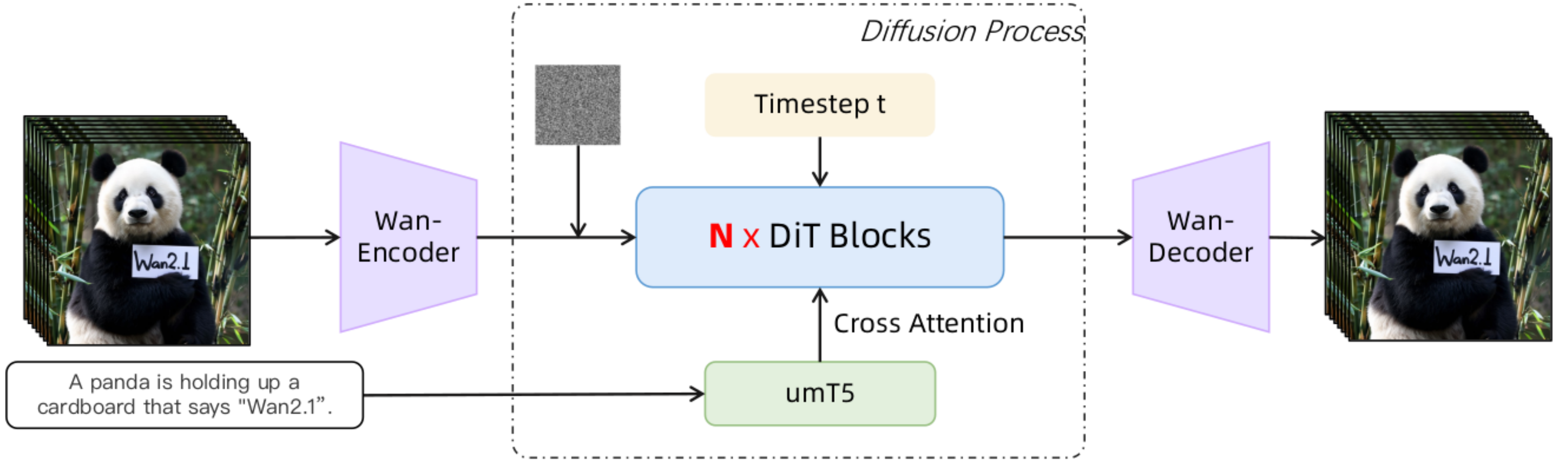
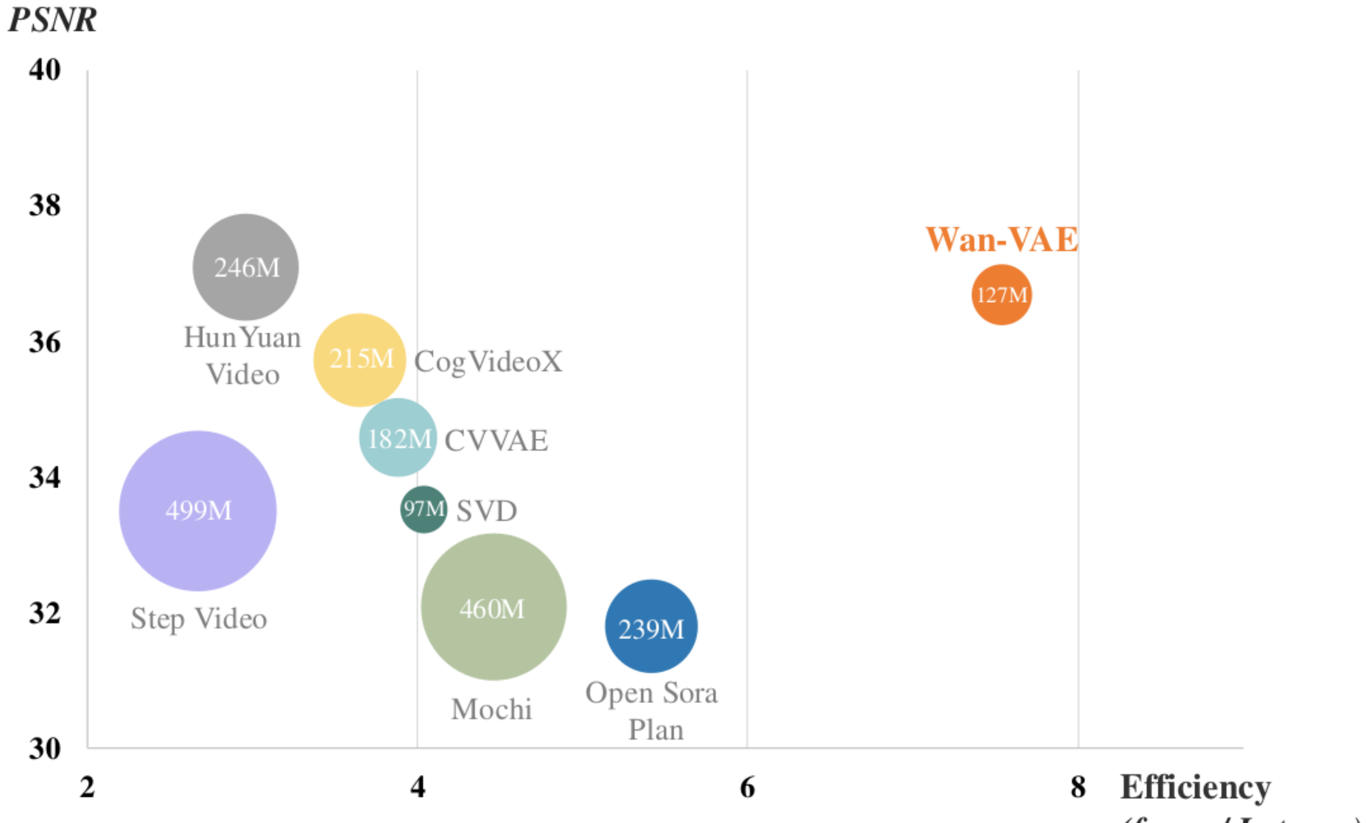


Figure 9: Architecture of the Wan.

- Wan-VAE: 3D causal VAE through feature cache mechanism
- Diffusion Transformer (DiT) model architecture
- Cross-attention to embed the input text or image conditions

Wan: Open and Advanced Large-Scale Video Generative Models



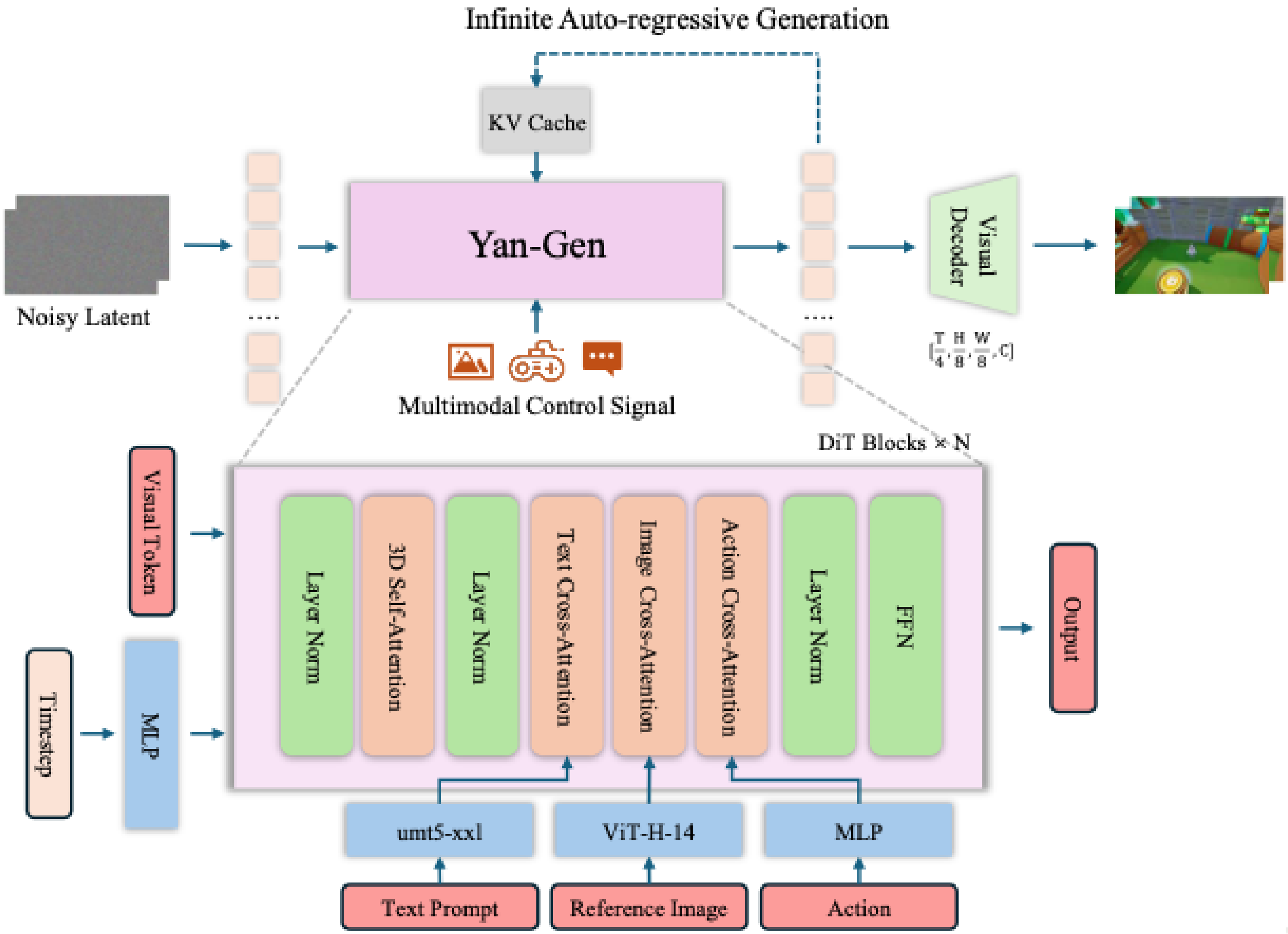
Note 1: The area of the sphere represents the size of the model parameters.
Note 2: The default compression rate is 4×8×8, except for SVD: 1×8×8, Mochi: 6×8×8, and Step Video: 8×16×16.

Wan -> Yan-Gen

- Anti-drifting against error accumulation and bias exposure
  - Hierarchical captioning
- Multimodal conditions such as image, text, and action
  - Decoupled cross-attention layers
  - Multimodal to video (MM2V) training
- Causal model
  - Autoregressive post-training
- Real-time interaction
  - Distillation through self-forcing post-training

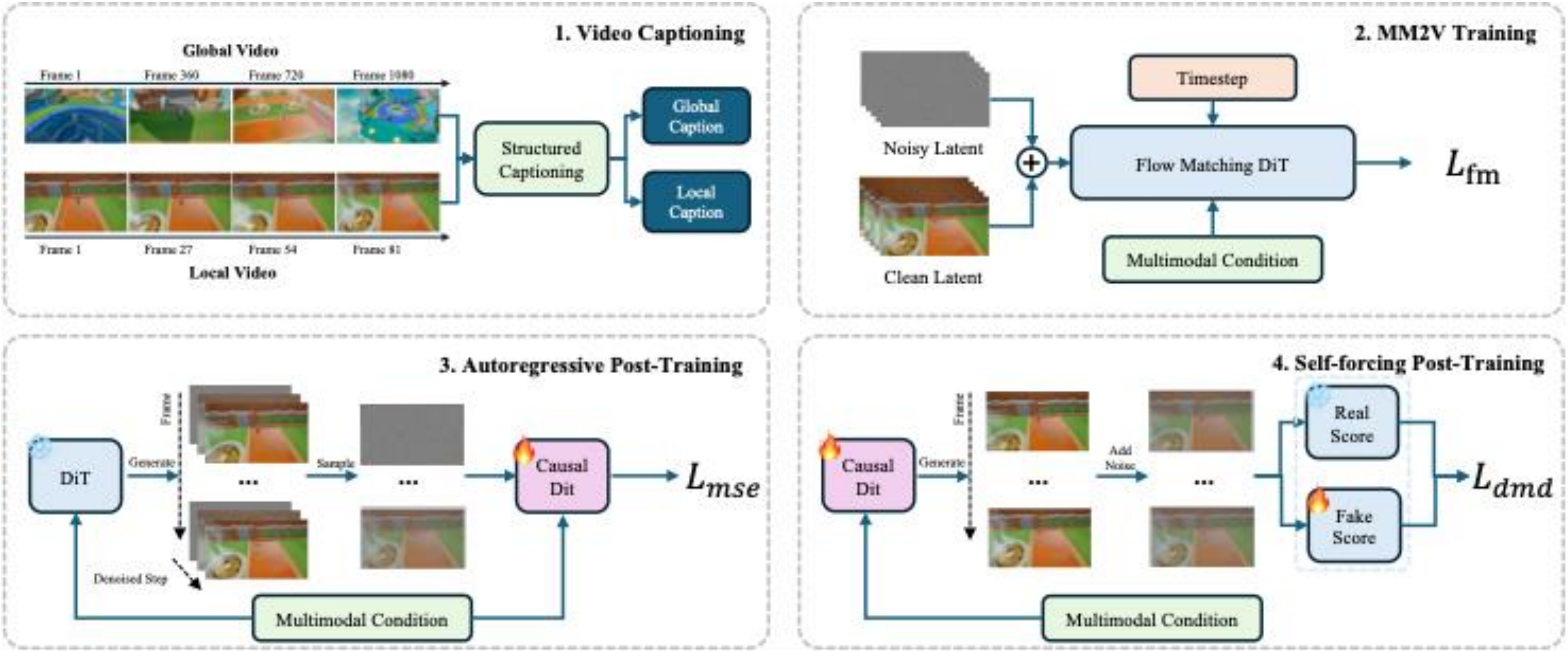
Yan-Gen: Architecture

- umt5-xxl text encoding (512 tokens)
- ViT-H-14 image embedding (257 tokens)
- Decoupled cross attention for text, image and action inputs



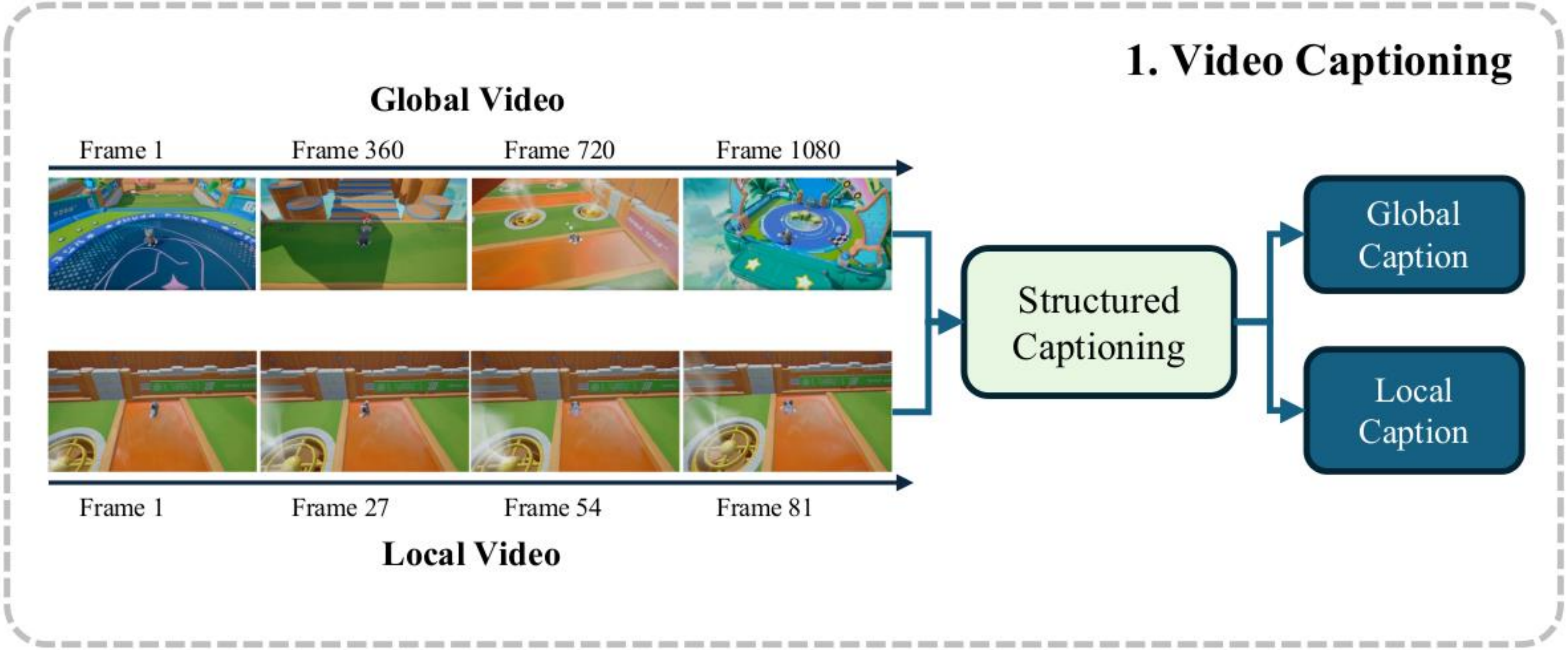
Georgia Tech

Yan-Gen: 4-Stage Training



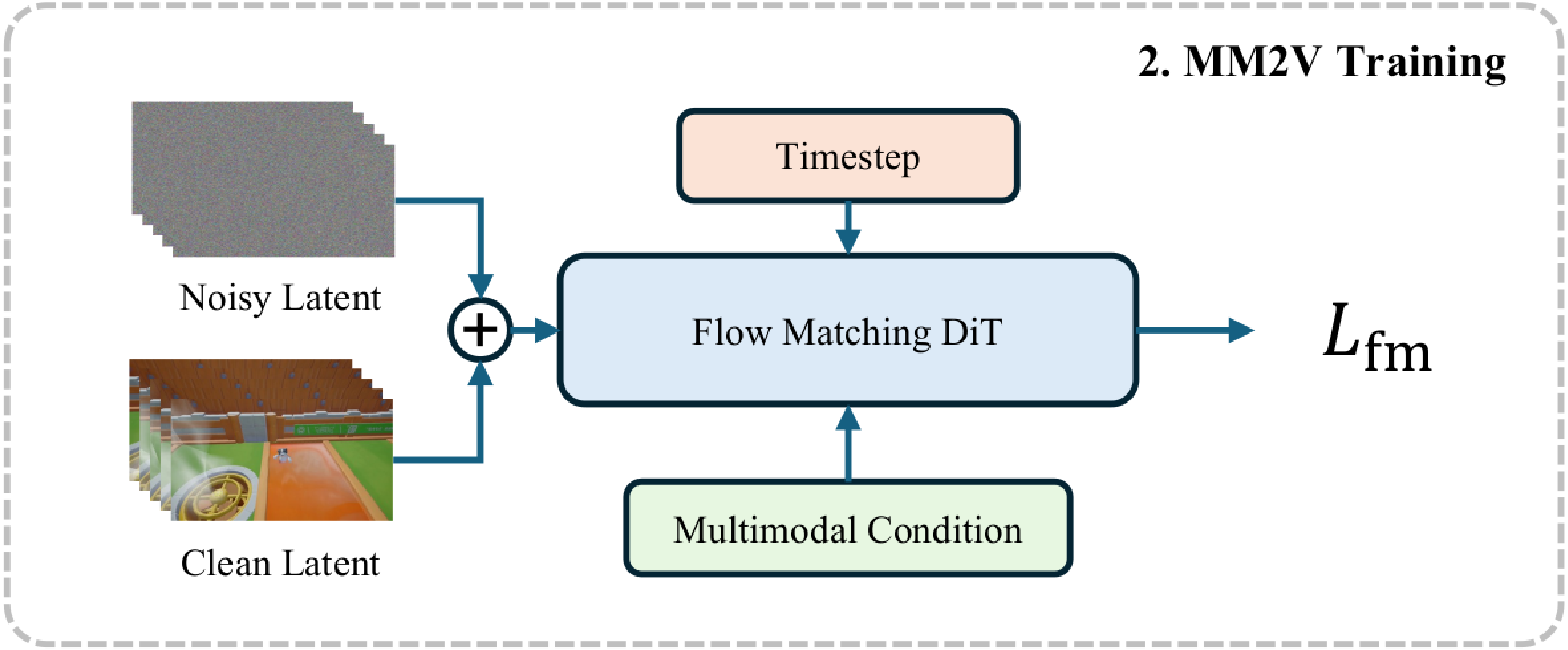
Georgia Tech

Yan-Gen: Training – 1) Video Captioning



- Hierarchical captioning on 98 million frames using Qwen2.5VL
- Global Captioning (Static and High-level Description)
  - Global layout, visual theme, base lighting and weather
- Local Captioning (Grounding Dynamic Events)
  - Local scene, interactive objects, and critical events

Yan-Gen: Training – 2) MM2V Training



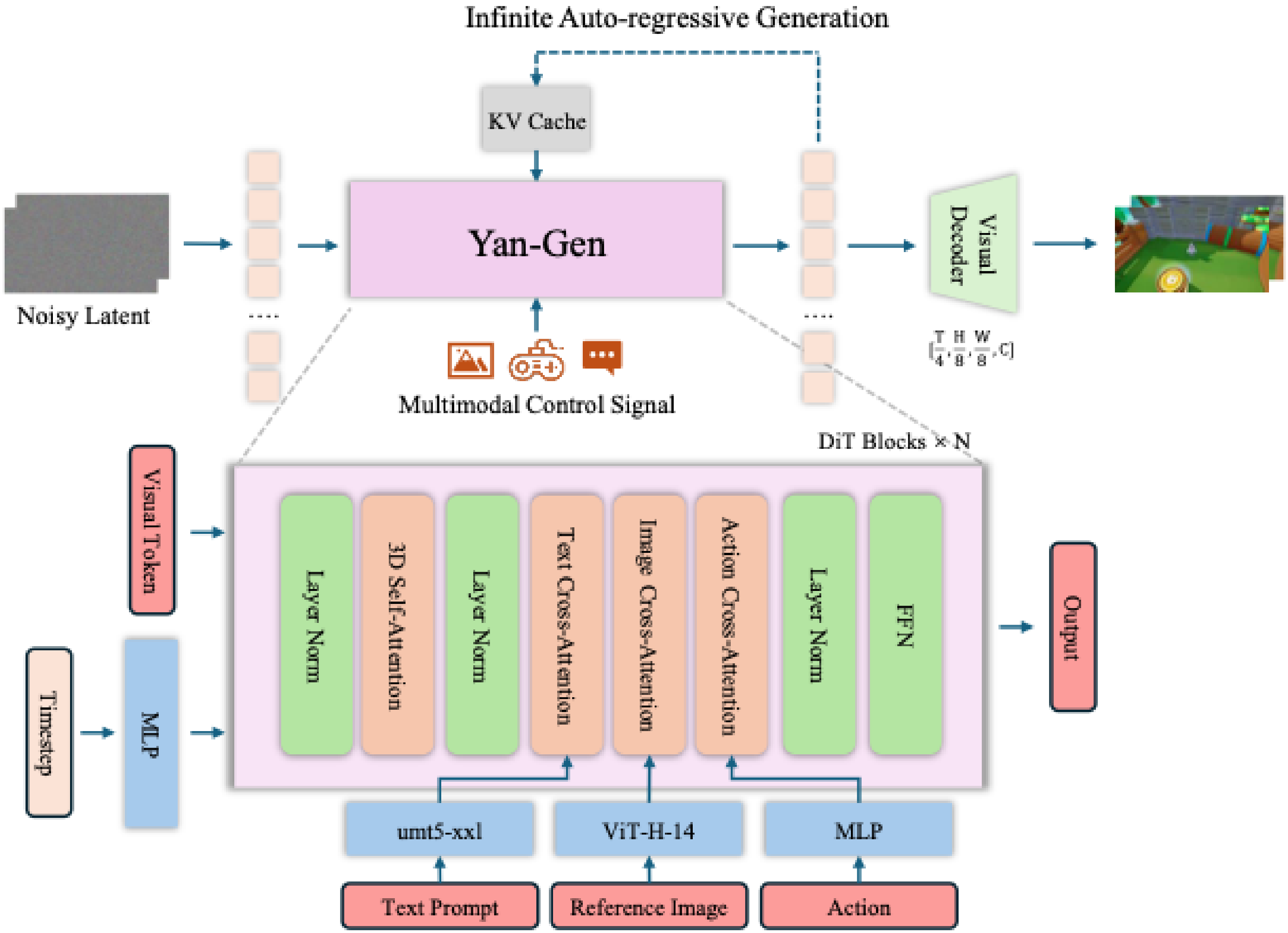
$$\mathcal{L}_{fm} = \mathbb{E}_{x_0, x_1, c_{txt}, c_{img}, c_{act}, t} || u(x_t, c_{txt}, c_{img}, c_{act}, t; \theta) - v_t ||^2$$

- Goal: Generate fixed-length videos guided by a reference image, descriptive text prompts(hierarchical caption + action description), and a sequence of user actions.

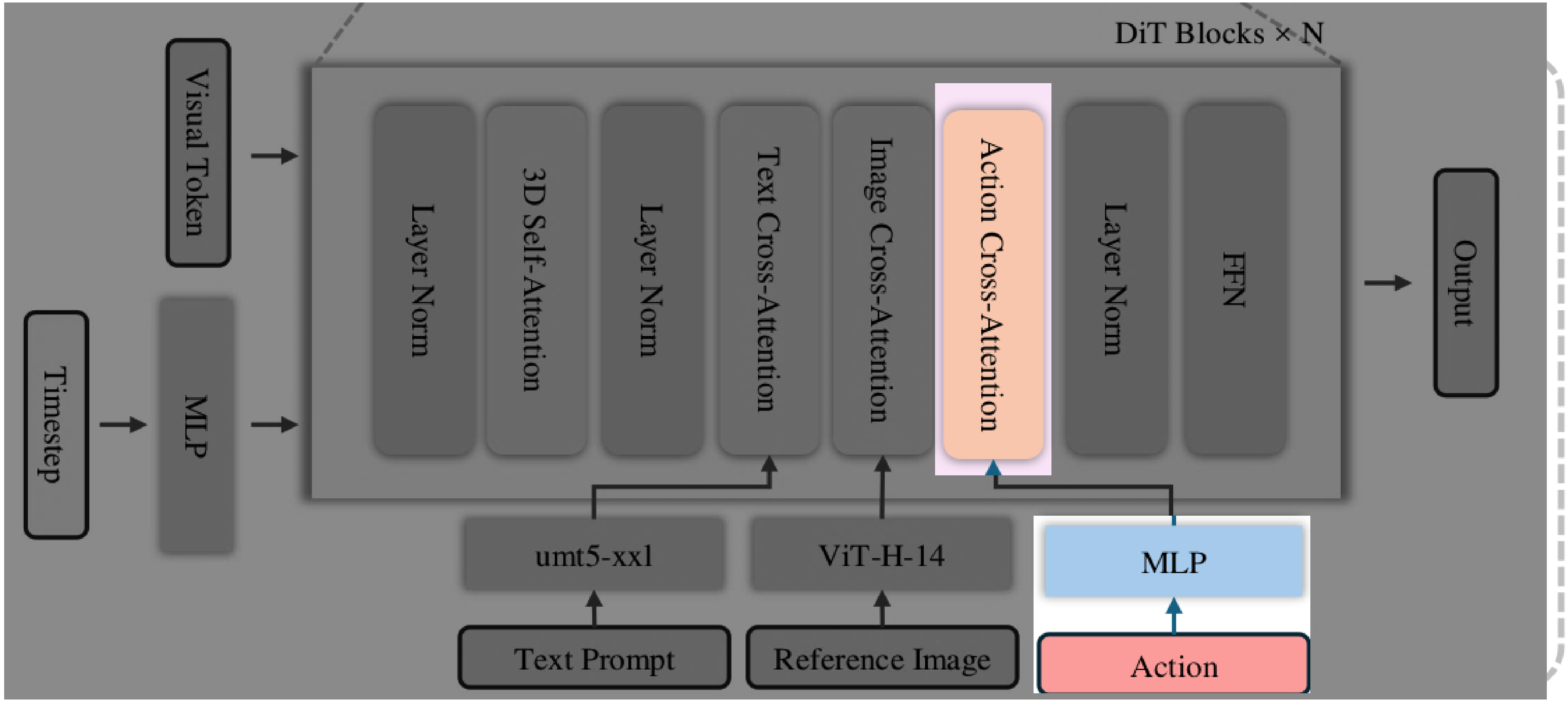
Yan-Gen: Training 2) MM2V - Adaptation Phase

- Finetuned the pretrained Wan model with Low-Rank Adaptation (LoRA) for image- and text-based video generation

- With p=0.1, model only have global caption



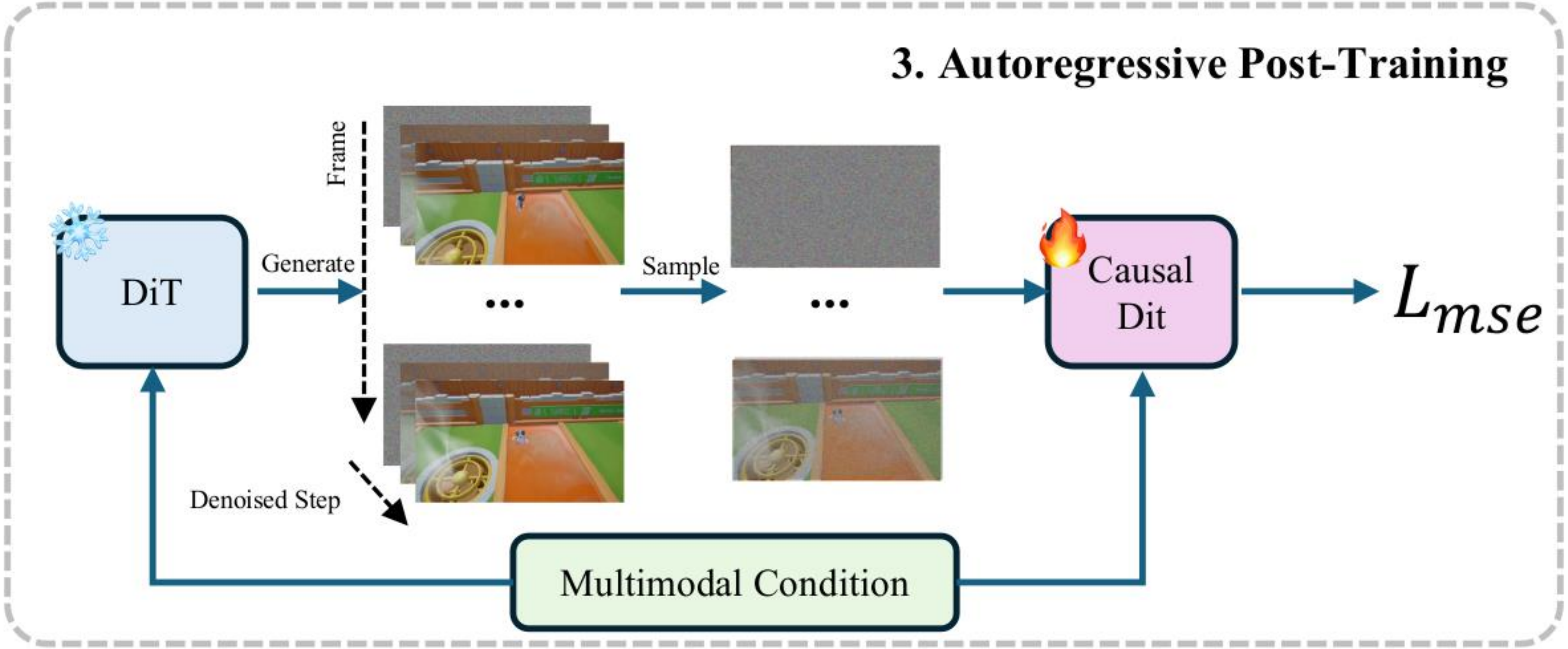
Yan-Gen: Training – 2) MM2V – Action Module



$$\mathcal{L}_{fm} = \mathbb{E}_{x_0,x_1,c_{txt},c_{img},c_{act},t}||u(x_t, c_{txt}, c_{img}, c_{act}, t; \theta) - v_t||^2$$

- Goal: Generate fixed-length videos guided by a reference image, descriptive text prompts(hierarchical caption + action description), and a sequence of user actions.

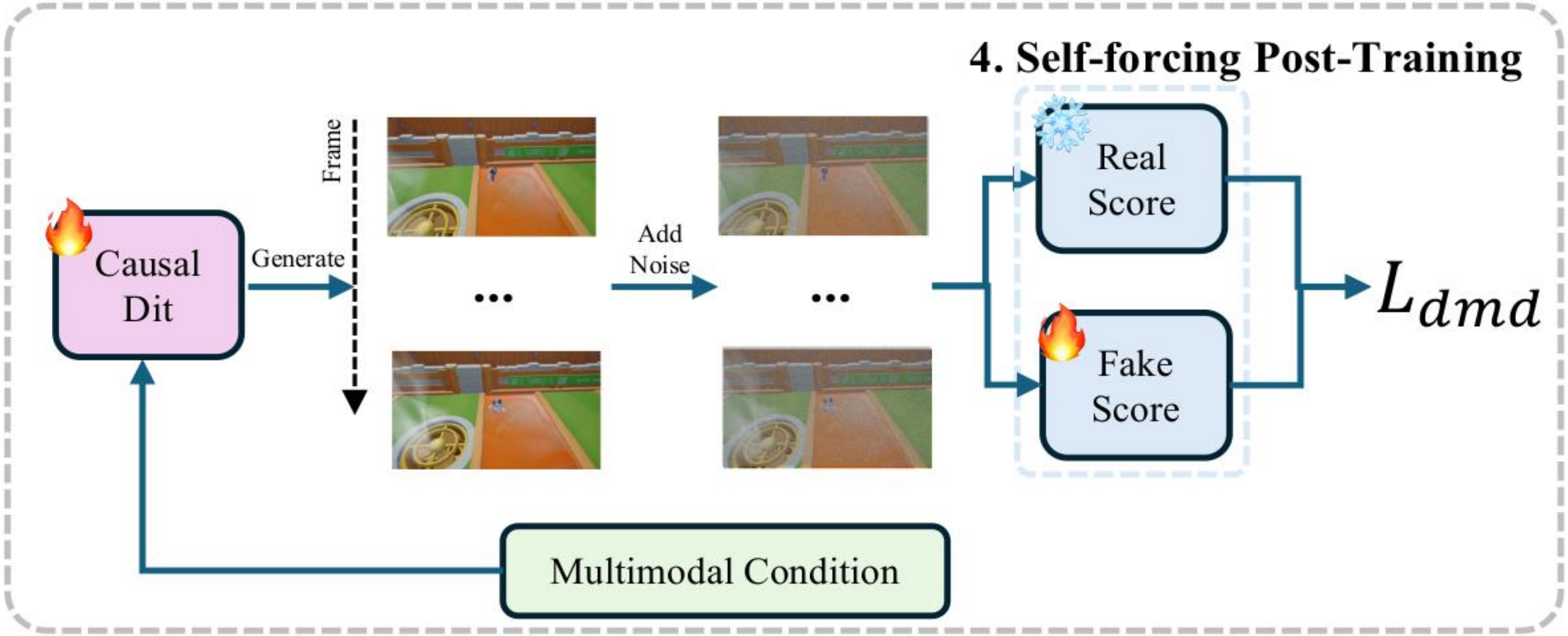
Yan-Gen: Training – 3) Autoregressive Post-Training



$$\mathcal{L}_{mse} = \mathbb{E}_{x,c_{txt},c_{img},c_{act},t^i} \left\| u'\left(\{x^i_{t^i}\}_{i=1}^N, \{t^i\}_{i=1}^N\right) - \{x^i_0\}_{i=1}^N \right\|^2$$

- Goal: Convert Yan-Gen into a causal and autoregressive model

Yan-Gen: Training – 4) Self-Forcing Post-Training

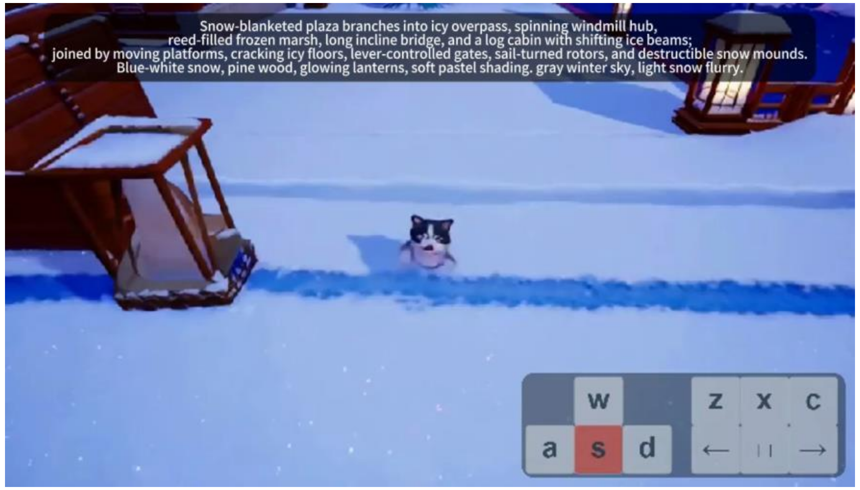


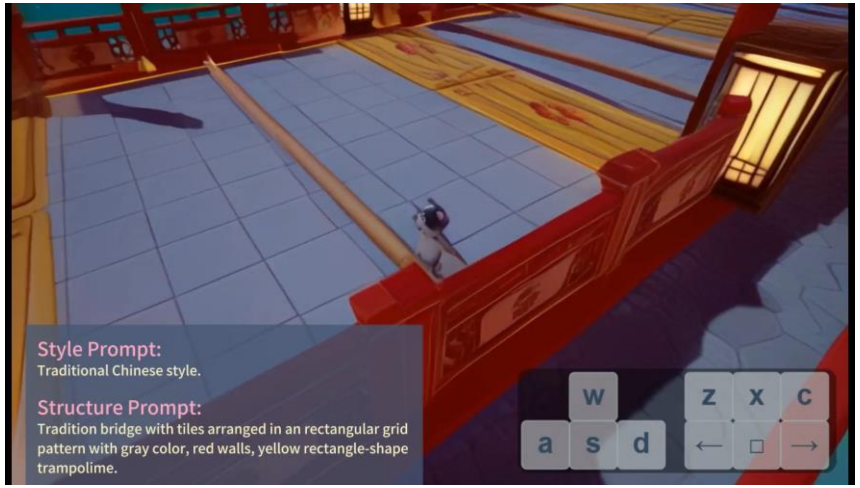
$$\nabla_\phi \mathcal{L}_{dmd} \triangleq \mathbb{E}_t \left(\nabla_\phi \mathbf{KL} \left(p_{\text{gen},t} \| p_{\text{data},t} \right) \right)$$

- Goal: Extracting a few-step generator using distribution matching distillation (DMD)

Yan-Gen: Results

- Text-to-Interactive Video
- Text-Guided Interactive Video Expansion
- Image-to-Interactive Video
- Multimodal Cross-Domain Fusion

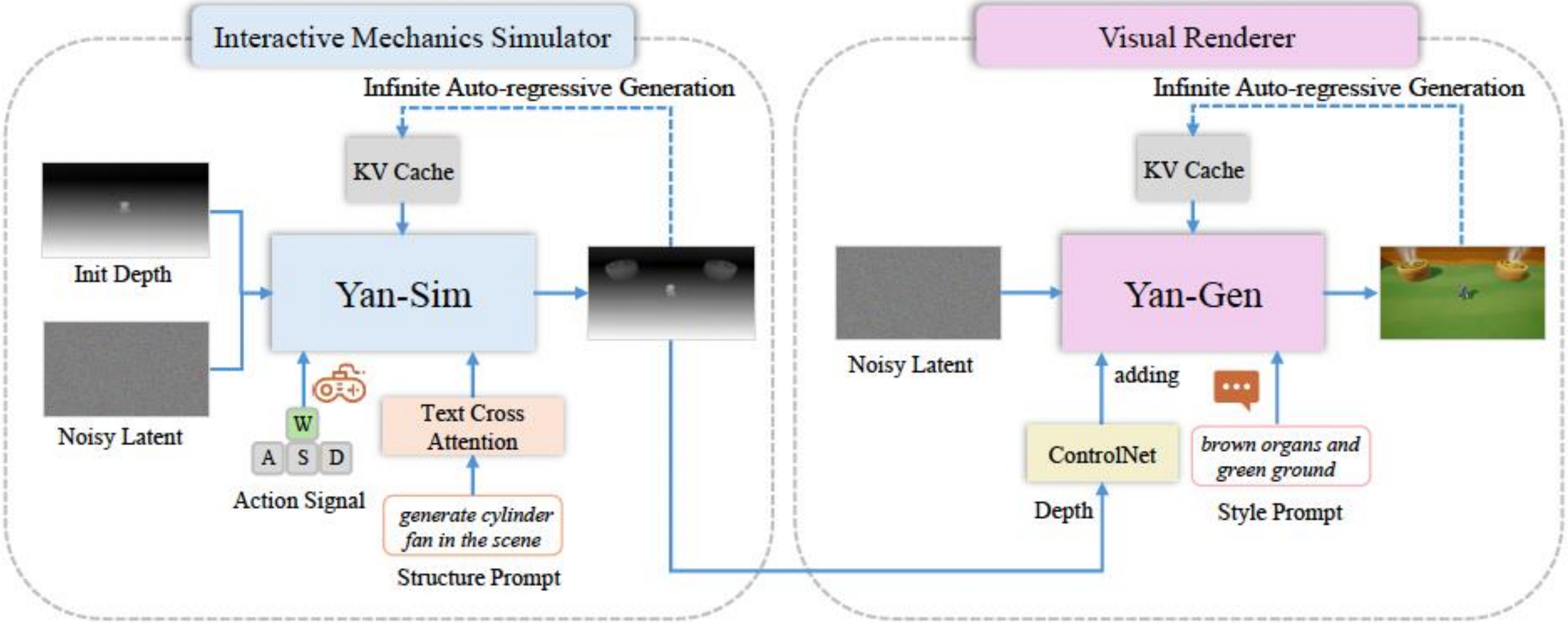




Yan Edit

- Real-time multi-granular video content editing using text
- Built on top of other Yan models
  - Interactive Mechanics Simulator (Yan-Sim)
  - Visual Renderer (Yan-Gen)

Yan Edit: Architecture



- Interactive Mechanics Simulator + Visual Renderer
  - Simulator – Maintains physics + interactivity
  - Renderer – Maintains style

Yan Edit: Training – Simulator (Yan-Sim)

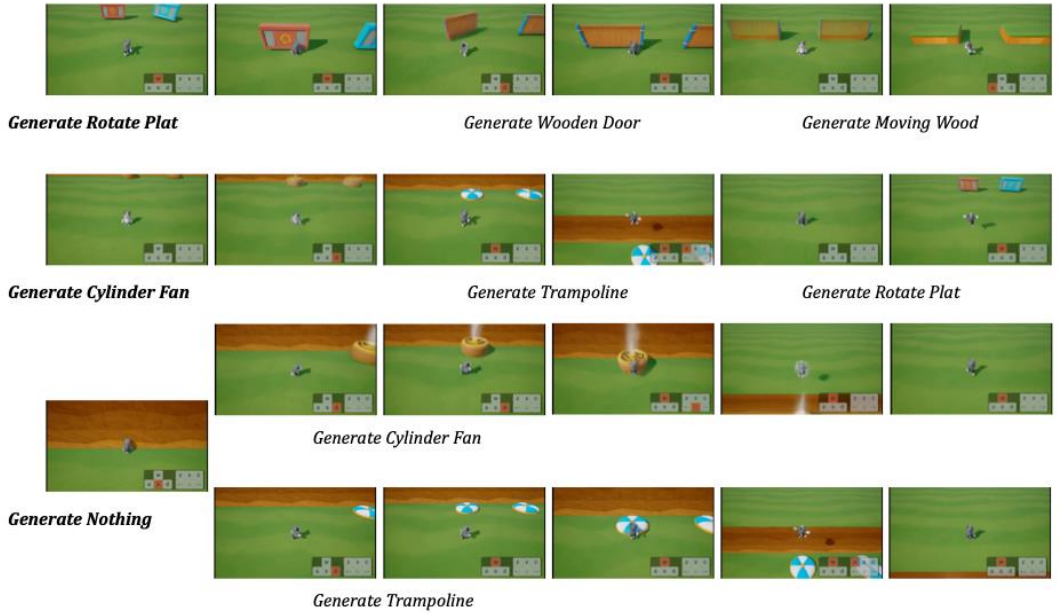
- Data preparation
  - Text Embeddings: Qwen2.5-VL to generate description for objects
  - Latent Space: VAE Training to compress depth maps into latent space
  - Action Embeddings: User controls
- Yan-Sim Training Phase 1:
  - Additional text cross-attention layer for text descriptions
  - Joint optimization between spatial, action, and text in UNet block
- Yan-Sim Training Phase 2:
  - Freeze text cross-attention after structural alignment with text descriptions
  - Fine-tune action layers for fine-grained details

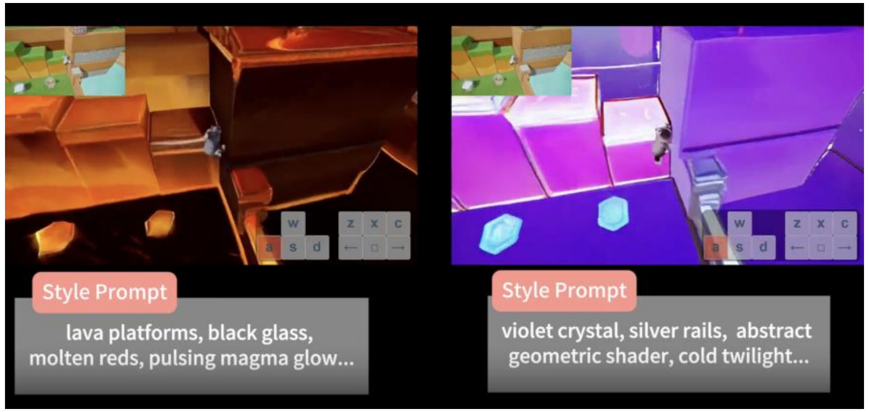
Yan Edit: Training - Renderer

- Data preparation
  - Depth map: Process maps from simulator trainings through ControlNet
  - Text: Style-based prompt
    - In-domain: Style captions from Yan dataset
    - Out-of-domain: New captions generated using GPT-4
- Yan-Gen 4 Stage Training
  - Video Captioning – Used as in-domain prompts
  - MM2V Training
  - Autoregressive Post Training – combined with ControlNet weights in DiT
  - Self-Forcing Post-Training

Yan Edit - Evaluation

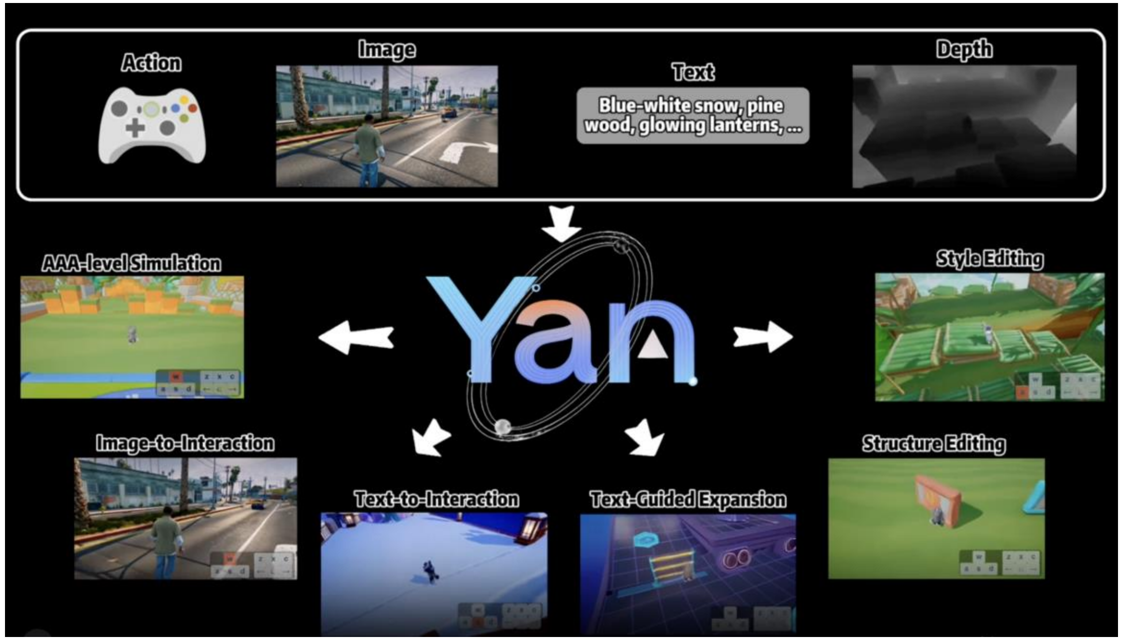
- Structure Editing
  - Dynamic structure prompts during interaction
  - Real-time content generation
- Style Editing
  - Dynamic rendering style changes
  - Open-domain style editing with accurate interactions





Strengths

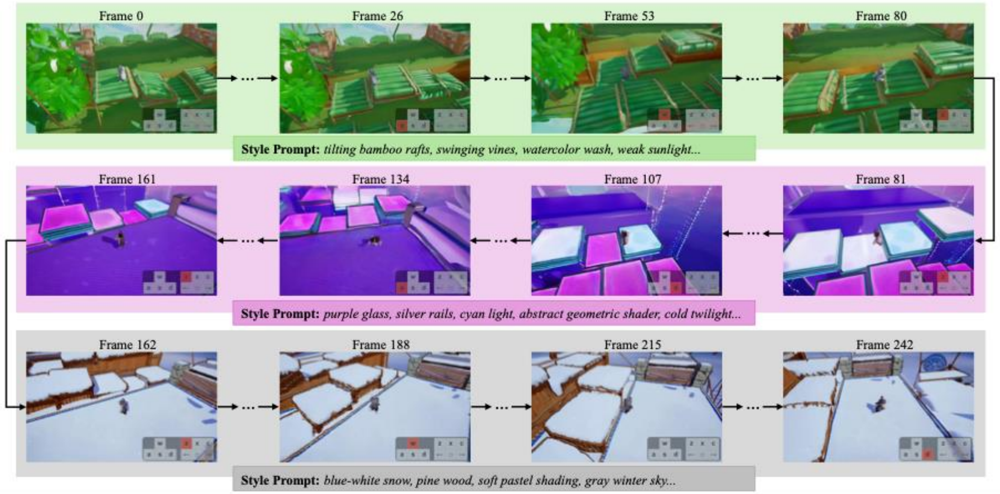
- Unified framework for interactive video generation
  - Simulation, generation, editing
- Open-domain interactive worlds
- Accurate resolution and physical world modeling
- Large scale dataset collection + generation



Georgia Tech

Limitations & Societal Implications

1. Visual consistency across long durations



2. Impractical real-world application

- Underlying game environment

3. Compute Inefficiency

- Uses A100 for inference

Discussion Points

- What may be some tradeoffs if we were to use real-world datasets instead of game-centric ones?

- Which applications would benefit from using virtual (i.e. game-based) datasets?

- Do you think user platforms (i.e. social media) benefit from AI content?

- Yan provides a unified framework for us to produce and edit interactive videos. Is an interactive model the future of video generation?

- Would you classify Yan as closer to a video generation model or world model?