Penn Engineering
University of Pennsylvania

GMLR

CIS 6280 · FALL 2026

WORLD MODELS

LECTURE 12

Video Diffusion Models

Jiatao Gu

University of Pennsylvania · October 15, 2026

Course announcements

Final project	Project information releases October 16.
Assignment 2	Due October 19.
Office hours	Thursday, 4:30–5:30 PM, AGH 423.

Diffusion as a hierarchical VAE

$$p_\theta(y_{0:J}) = p(y_J) \prod_{j=1}^{J} p_\theta(y_{j-1} \mid y_j)$$

Inference distribution q	Generative model p
Assign noisy latent variables to a known clean image	Generate an image through learned reverse transitions

$$\mathcal{L}_j = \mathbb{E}_q[D_{\mathrm{KL}}(q(y_{j-1} \mid y_j, y_0) \,\|\, p_\theta(y_{j-1} \mid y_j))]$$

This is a middle term of the negative ELBO. y_0 is known when defining the training target.

How does the choice of q determine the reverse target?

Ho et al., DDPM (2020)

The DDPM forward process

$$q(y_{1:J} \mid y_0) = \prod_{j=1}^{J} q(y_j \mid y_{j-1})$$

$$q(y_j \mid y_{j-1}) = \mathcal{N}\left(\sqrt{\alpha_j}\, y_{j-1},\, \beta_j I\right)$$

$\alpha_j = 1 - \beta_j$. The fixed schedule sets how much signal survives each step.

$$q(y_j \mid y_0) = \mathcal{N}\left(\sqrt{\bar{\alpha}_j}\, y_0,\, (1 - \bar{\alpha}_j) I\right), \qquad \bar{\alpha}_j = \prod_{i=1}^{j} \alpha_i$$

Given the previous noisy image, the next corruption step needs no earlier image.

Deriving the DDPM reverse target

During training, the clean image y_0 is known. For $j \geq 2$:

$$q(y_{j-1} \mid y_j, y_0) \propto q(y_j \mid y_{j-1})\, q(y_{j-1} \mid y_0)$$

Write $u = y_{j-1}$. Multiplying the two Gaussians gives:

$$\log q(u \mid y_j, y_0) = -\frac{1}{2}\left[\frac{\|y_j - \sqrt{\alpha_j}u\|^2}{\beta_j} + \frac{\|u - \sqrt{\bar{\alpha}_{j-1}}y_0\|^2}{1-\bar{\alpha}_{j-1}}\right] + \text{const}$$

Collect the scalar quadratic coefficient A_j and vector linear coefficient B_j:

$$= -\frac{A_j}{2}\left\|u - \frac{B_j}{A_j}\right\|^2 + \text{const}$$

Complete the square: the reverse target has mean B_j/A_j and covariance $A_j^{-1}I$.

Ho et al., DDPM (2020)

The DDPM reverse mean and variance

$$A_j = \frac{\alpha_j}{\beta_j} + \frac{1}{1-\bar{\alpha}_{j-1}}, \qquad B_j = \frac{\sqrt{\alpha_j}}{\beta_j} y_j + \frac{\sqrt{\bar{\alpha}_{j-1}}}{1-\bar{\alpha}_{j-1}} y_0$$

$$q(y_{j-1} \mid y_j, y_0) = \mathcal{N}(\tilde{\mu}_j, \tilde{\beta}_j I)$$

$$\tilde{\mu}_j = \frac{B_j}{A_j} = \frac{\sqrt{\bar{\alpha}_{j-1}}\beta_j}{1-\bar{\alpha}_j} y_0 + \frac{\sqrt{\alpha_j}(1-\bar{\alpha}_{j-1})}{1-\bar{\alpha}_j} y_j$$

$$\tilde{\beta}_j = A_j^{-1} = \frac{1-\bar{\alpha}_{j-1}}{1-\bar{\alpha}_j}\,\beta_j$$

The chosen forward process fixes this clean-conditioned reverse target.

The learned DDPM reverse step

$$\hat{\epsilon} = \epsilon_\theta(y_j, j), \qquad \hat{y}_0 = \frac{y_j - \sqrt{1-\bar{\alpha}_j}\,\hat{\epsilon}}{\sqrt{\bar{\alpha}_j}}$$

Use the clean estimate in the posterior mean:

$$\mu_\theta(y_j, j) = \tilde{\mu}_j(y_j, \hat{y}_0) = \frac{1}{\sqrt{\alpha_j}}\left(y_j - \frac{\beta_j}{\sqrt{1-\bar{\alpha}_j}}\hat{\epsilon}\right)$$

$$y_{j-1} = \mu_\theta(y_j, j) + \sqrt{\tilde{\beta}_j}\,\xi, \qquad \xi \sim \mathcal{N}(0, I)$$

This display uses the posterior-variance choice, for $j \geq 2$.

The same noise levels, different connections

$$a_j = \sqrt{\bar{\alpha}_j}, \qquad b_j = \sqrt{1 - \bar{\alpha}_j}$$

$$q(y_j \mid y_0) = \mathcal{N}(a_j y_0, b_j^2 I), \qquad y_j = a_j y_0 + b_j \epsilon$$

Denoising training samples a clean image, one noise level, and one noise draw.

These single-level distributions do not uniquely specify how different levels connect.

DDIM keeps the noisy marginals and constructs a different joint inference distribution.

Constructing a new connection between levels

With the clean training image known, extract the current noise:

$$\epsilon_j = \frac{y_j - a_j y_0}{b_j}, \qquad \epsilon_j \mid y_0 \sim \mathcal{N}(0, I)$$

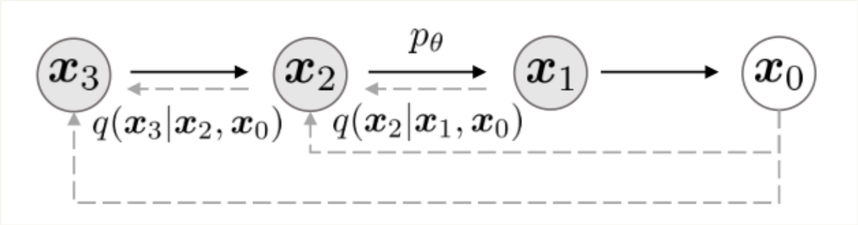
Connect the previous level using this noise and an independent draw $\xi \sim \mathcal{N}(0, I)$:

$$y_{j-1} = a_{j-1} y_0 + c_j \epsilon_j + \sigma_j \xi$$

$$\underbrace{c_j^2 + \sigma_j^2}_{\text{noise variance at the previous level}} = b_{j-1}^2 \quad \Longrightarrow \quad c_j = \sqrt{b_{j-1}^2 - \sigma_j^2}$$

Choose how much noise to retain and how much fresh noise to add.

The non-Markovian inference process



DDIM, Figure 1, right. Dashed inference paths also use the clean data.

$$q_\sigma(y_{1:J} \mid y_0) = q(y_J \mid y_0) \prod_{j=2}^{J} q_\sigma(y_{j-1} \mid y_j, y_0)$$

$$q_\sigma(y_j \mid y_{j-1}, y_0) = \frac{q_\sigma(y_{j-1} \mid y_j, y_0)\, q(y_j \mid y_0)}{q(y_{j-1} \mid y_0)}$$

The chronological forward step generally still depends on y_0, even after y_{j-1} is known.

This defines q, the inference distribution. It is not yet the learned generator p.

The new reverse target and the learned model

$$q_\sigma(y_{j-1} \mid y_j, y_0) = \mathcal{N}\big(m_{\sigma,j}(y_j, y_0),\, \sigma_j^2 I\big)$$

$$m_{\sigma,j} = a_{j-1} y_0 + \frac{\sqrt{b_{j-1}^2 - \sigma_j^2}}{b_j}(y_j - a_j y_0)$$

Generation has no known clean image. Predict it from the current sample:

$$\hat{\epsilon} = \epsilon_\theta(y_j, j), \qquad \hat{y}_0 = \frac{y_j - b_j \hat{\epsilon}}{a_j}$$

$$p_{\theta,\sigma}(y_{j-1} \mid y_j) := q_\sigma(y_{j-1} \mid y_j, \hat{y}_0(y_j))$$

Deriving the generalized reverse update

Sample the Gaussian model from the previous page:

$$y_{j-1} = a_{j-1}\hat{y}_0 + \frac{\sqrt{b_{j-1}^2 - \sigma_j^2}}{b_j}(y_j - a_j\hat{y}_0) + \sigma_j\xi$$

The clean estimate gives $y_j - a_j\hat{y}_0 = b_j\hat{\epsilon}$. Therefore:

$$y_{j-1} = \underbrace{a_{j-1}\hat{y}_0}_{\text{clean signal}} + \underbrace{\sqrt{b_{j-1}^2 - \sigma_j^2}\,\hat{\epsilon}}_{\text{retained noise}} + \underbrace{\sigma_j\xi}_{\text{fresh noise}}$$

The same noisy training examples support the same noise-regression surrogate.

DDPM as a particular variance choice

$$\sigma_j^2 = \tilde{\beta}_j = \frac{1 - \bar{\alpha}_{j-1}}{1 - \bar{\alpha}_j}\,\beta_j$$

$$\frac{\sqrt{b_{j-1}^2 - \tilde{\beta}_j}}{b_j} = \frac{\sqrt{\alpha_j}(1 - \bar{\alpha}_{j-1})}{1 - \bar{\alpha}_j}$$

Substitution recovers the DDPM posterior mean coefficients and its chosen variance.

Variance choice	Resulting sampling rule
$\sigma_j^2 = \tilde{\beta}_j$	The posterior-variance DDPM used in our derivation
$\sigma_j = 0$	A deterministic update, derived next

DDIM generalizes the inference construction underlying the sampler.

Song et al., DDIM (2020) · Ho et al., DDPM (2020)

Deterministic DDIM sampling

Set $\sigma_j = 0$ in the generalized update:

$$y_{j-1} = a_{j-1}\hat{y}_0 + b_{j-1}\hat{\epsilon}$$

$$\hat{\epsilon} = \epsilon_\theta(y_j, j), \qquad \hat{y}_0 = \frac{y_j - b_j\hat{\epsilon}}{a_j}$$

What remains random	What happens after initialization
The initial Gaussian sample y_J	Each next sample follows from the current network prediction

Selected levels can also be connected as $y_k = a_k\hat{y}_0 + b_k\hat{\epsilon}$, for $k < j$.

Re-evaluate the denoiser after each update. Fewer selected levels can reduce sampling cost.

DDIM as an Euler update

Substitute the clean estimate into deterministic DDIM and divide by a_k:

$$\frac{y_k}{a_k} = \frac{y_j}{a_j} + \left(\frac{b_k}{a_k} - \frac{b_j}{a_j}\right)\hat{\epsilon}$$

Use a rescaled sample $u_j = y_j/a_j$, and a noise-to-signal coordinate $s_j = b_j/a_j$.

$$\underbrace{u_k}_{\text{next value}} = \underbrace{u_j}_{\text{current value}} + \underbrace{(s_k - s_j)}_{\text{coordinate step}}\ \underbrace{\epsilon_\theta(a_j u_j, j)}_{\text{predicted direction}}$$

$$\frac{du}{ds} = \epsilon_\theta\left(\frac{u}{\sqrt{1+s^2}}, j(s)\right)$$

Continuous view: $j(s)$ denotes the corresponding noise level. Generation moves toward smaller s.

Song et al., DDIM, §4.3

Generation with an ODE

An ordinary differential equation (ODE) specifies the local velocity of a sample.

$$\frac{dr_\tau}{d\tau} = v_\theta(r_\tau, \tau), \qquad r_0 \sim \mathcal{N}(0, I)$$

Here τ increases from noise at 0 to data at 1. The network v_θ predicts a velocity.

$$r_{\tau+\Delta\tau} \approx r_\tau + \Delta\tau\, v_\theta(r_\tau, \tau)$$

Euler sampling	Evaluate the velocity at the current sample, take a short step, and evaluate again.
Starting noise	Different initial samples can follow different deterministic trajectories.

A stochastic differential equation (SDE) can also inject random perturbations during evolution.

Song et al., DDIM, §4.3 · Lipman et al., Flow Matching (2023)

A path between noise and data

$$n \sim \mathcal{N}(0, I), \qquad y_0 \sim p_{\text{data}}$$

$$r_\tau = (1-\tau)n + \tau y_0, \qquad 0 \le \tau \le 1$$

$$\frac{dr_\tau}{d\tau} = y_0 - n$$

Generation coordinate	Endpoint
$\tau = 0$	Gaussian noise
$\tau = 1$	Data

Lipman et al., Flow Matching (2023)

The flow matching objective

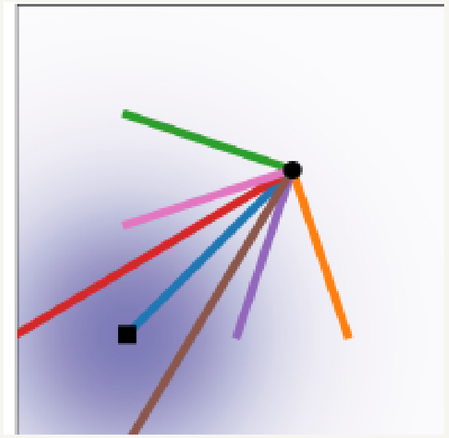
$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}_{n,y_0,\tau}\left[\left\|v_\theta(r_\tau,\tau) - (y_0 - n)\right\|^2\right]$$

Input	r_τ and the coordinate τ
Prediction	The velocity v_θ
Target	The known pair velocity $y_0 - n$

Training uses a sampled point on the path, without running the sampler.

Lipman et al., Flow Matching (2023)

Pair velocities and the learned field



Straight conditional trajectories, excerpt from the Flow Matching paper.

Different training pairs can produce similar intermediate inputs.

The regression target can differ between those pairs.

The network learns a field that averages the compatible velocities.

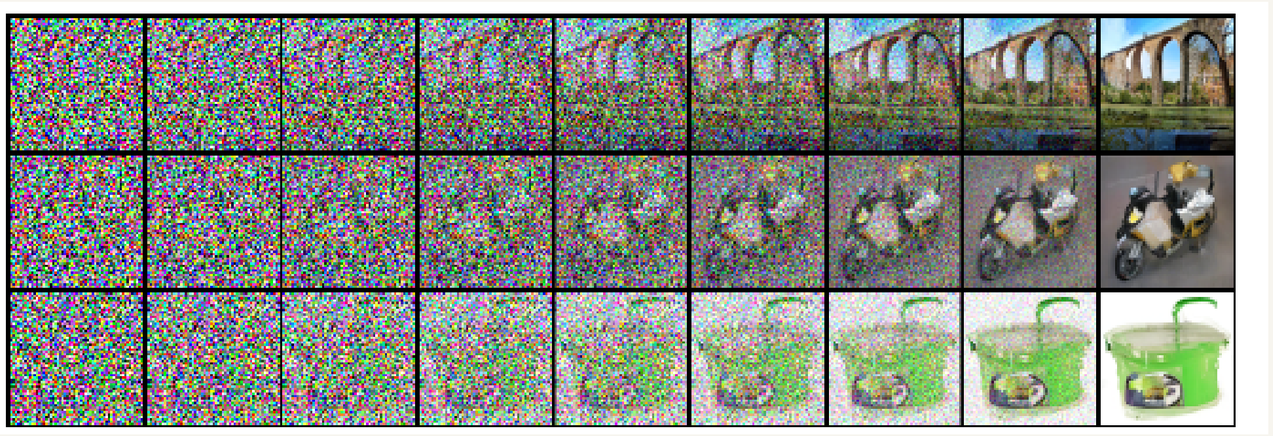
$$v^*(r, \tau) = \mathbb{E}[y_0 - n \mid r_\tau = r]$$

A straight training path does not require a straight generated trajectory.

Sampling with a velocity field

$$r_0 \sim \mathcal{N}(0, I), \qquad \frac{dr_\tau}{d\tau} = v_\theta(r_\tau, \tau)$$

$$r_{\tau+\Delta\tau} \approx r_\tau + \Delta\tau\, v_\theta(r_\tau, \tau)$$



Flow Matching paper: samples along a generated trajectory.

Euler integration: predict the local direction, take a step, and evaluate again.

Conditioning the generator

$$p_\theta(y_0 \mid c), \qquad \epsilon_\theta(y_j, j, c)$$

Condition	How the network receives it
Text	Read text-encoder embeddings through attention
Reference image	Read encoded visual evidence

Train on examples paired with the supplied condition.

Conditioning defines a learned distribution for the chosen input.

Why use guidance?



Classifier-Free Guidance, Figure 1: increasing guidance from left to right for the same class.

Ordinary conditional sampling includes many possible outcomes.

Guidance can emphasize samples that express the chosen condition more strongly.

Sampling can trade variety against condition fidelity.

Conditional directions from Bayes' rule

$$p_j(y \mid c) = \frac{p_j(y)\, p_j(c \mid y)}{p(c)}$$

$$\nabla_y \log p_j(y \mid c) = \nabla_y \log p_j(y) + \nabla_y \log p_j(c \mid y)$$

Direction	Meaning
$\nabla_y \log p_j(y)$	The noisy-data score: move toward higher noisy-data density
$\nabla_y \log p_j(c \mid y)$	The change that makes the chosen condition more likely

Here y is a noisy sample at a fixed level j.

Classifier guidance

$$s_\theta(y_j, j) \approx \nabla_{y_j} \log p_j(y_j) \approx -\frac{\epsilon_\theta(y_j, j)}{b_j}$$

$$g_j = \nabla_{y_j} \log p_\phi(c \mid y_j, j)$$

$$s_{\text{guided}} = s_\theta + w g_j \quad \Longleftrightarrow \quad \epsilon_{\text{guided}} = \epsilon_\theta - w b_j g_j$$

p_ϕ is the classifier; w sets the guidance strength.

Train the classifier on noisy inputs and their noise levels.

The classifier gradient changes the prediction used by the sampler.

Dhariwal and Nichol, Classifier Guidance (2021) · Ho and Salimans, Classifier-Free Guidance (2022)

Classifier-free guidance: training

$$\epsilon_c = \epsilon_\theta(y_j, j, c), \qquad \epsilon_\varnothing = \epsilon_\theta(y_j, j, \varnothing)$$

With the condition	Train the conditional noise prediction
With condition dropout	Train the same network with a null condition
At sampling time	Obtain both predictions without a separate classifier

Two predictions from one jointly trained network replace the auxiliary classifier.

Ho and Salimans, Classifier-Free Guidance (2022)

Classifier-free guidance: the combination

$$\epsilon_{\text{guided}} = \epsilon_{\varnothing} + w\,(\epsilon_c - \epsilon_{\varnothing})$$

$$s_c - s_{\varnothing} \approx \nabla_{y_j} \log p_j(c \mid y_j)$$

Total scale w	Prediction
$w = 0$	Unconditional
$w = 1$	Ordinary conditional
$w > 1$	Extrapolate beyond the conditional prediction

Also written as $\epsilon_c + \gamma(\epsilon_c - \epsilon_{\varnothing})$, with $\gamma = w - 1$.

Ho and Salimans, Classifier-Free Guidance (2022)

Guidance inside the sampler

Predict twice	Use the same noisy sample and level, with and without the condition
Combine	Compute ϵ_{guided}
DDIM update	Use that prediction in both the clean estimate and the retained direction

$$\hat{y}_0^{\,g} = \frac{y_j - b_j \epsilon_{\text{guided}}}{a_j}, \qquad y_k = a_k \hat{y}_0^{\,g} + b_k \epsilon_{\text{guided}}$$

For a flow sampler: $v_{\text{guided}} = v_{\varnothing} + w(v_c - v_{\varnothing})$.

Song et al., DDIM (2020) · Ho and Salimans, Classifier-Free Guidance (2022) · Wan Team, Wan (2025)

Guidance strength and its limits



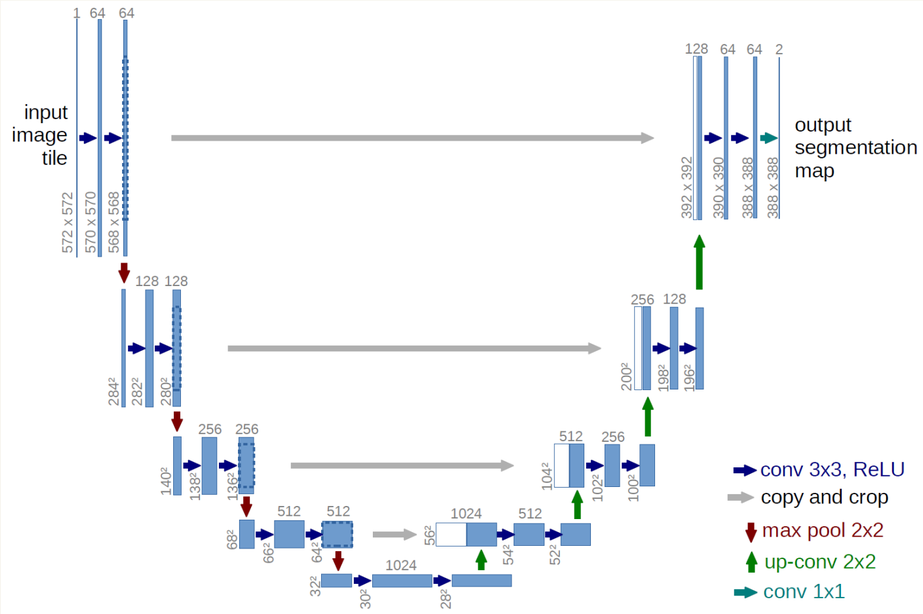
CFG paper, Figure 3, excerpt: ordinary conditional samples on the left; stronger guidance on the right.

Stronger guidance can favor	What it can lose
More typical, condition-aligned samples	Variety and coverage
Higher apparent fidelity	Natural colors or other details if the scale is excessive

Guidance strengthens a learned preference; it does not guarantee correct physical dynamics.

Ho and Salimans, Classifier-Free Guidance (2022)

U-Net denoisers



Original segmentation U-Net. Diffusion models adapt this multiscale structure.

Downsampling

Build features with a larger spatial context

Upsampling

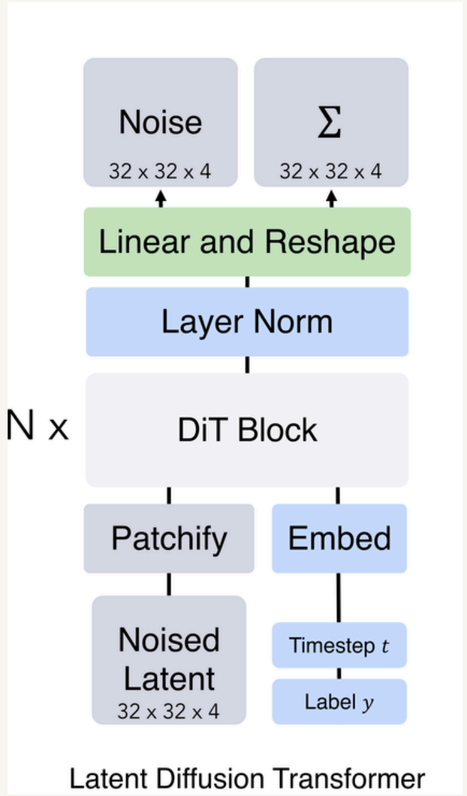
Recover a high-resolution dense prediction

Skip connections

Carry detailed features between matching resolutions

A diffusion U-Net also receives the noise level and condition, and predicts a denoising quantity.

Transformer denoisers



Original DiT architecture. This implementation uses a latent grid.

Divide the noisy tensor into patches.

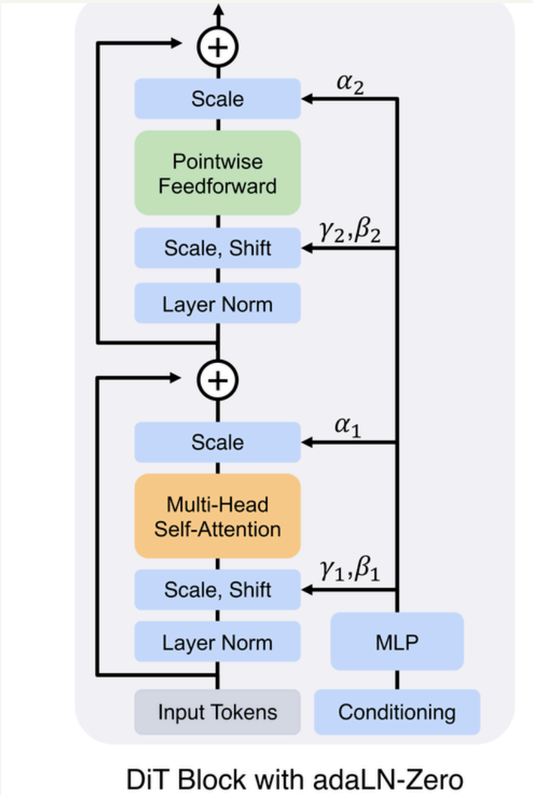
$$\mathrm{PatchEmbed}(\text{noisy input}) = (u_1, \ldots, u_N)$$

Each u_i is a continuous feature vector with a spatial position.

The transformer exchanges information between these patches.

Peebles and Xie, DiT (2023)

Inside a DiT block



DiT, Figure 3: adaptive layer-normalization conditioning.

Self-attention

Connect visual patches across the candidate image

Feedforward layers

Transform each token representation

Condition modulation

Supply the noise level and class information throughout the network

Peebles and Xie, DiT (2023)

A dense prediction from patch tokens

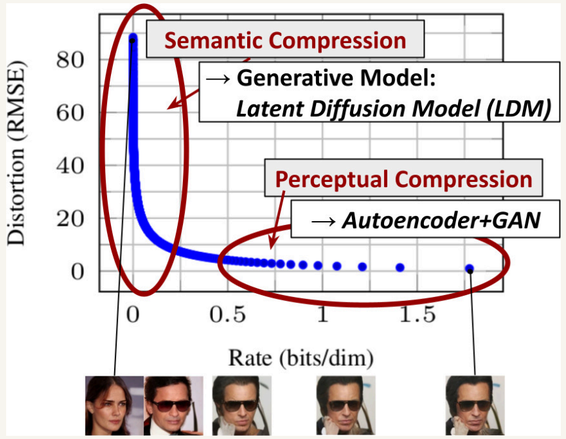
$$(u_1, \ldots, u_N) \xrightarrow{\text{Transformer}} (\hat{u}_1, \ldots, \hat{u}_N)$$

$$\text{Unpatchify}(\hat{u}_1, \ldots, \hat{u}_N) = \hat{\epsilon} \quad \text{or} \quad \hat{v}$$

The prediction has the same spatial organization as the tensor the sampler updates.

Backbone architecture and generation space are separate choices.

The space in which generation happens



Latent Diffusion paper: separating perceptual compression from repeated generative computation.

A denoiser can process raw pixels or an encoded representation.

Compression reduces repeated spatial computation, but the decoder must reconstruct what the representation preserves.

Rombach et al., Latent Diffusion (2022)

Latent diffusion: the reconstruction stage

$$z_0 = E_\phi(y_0), \qquad \tilde{y}_0 = D_\psi(z_0)$$

Module	Training role
Encoder E_ϕ	Map an observed image into a spatial latent grid
Decoder D_ψ	Reconstruct its visible detail from that grid

Train the reconstruction system first. Then freeze it while learning the latent generator.

The LDM paper studies perceptual autoencoders with mild KL or vector-quantization regularization.

Rombach et al., Latent Diffusion (2022)

Latent diffusion: the generative stage

$$z_0 = E_\phi(y_0), \qquad z_j = a_j z_0 + b_j \epsilon$$

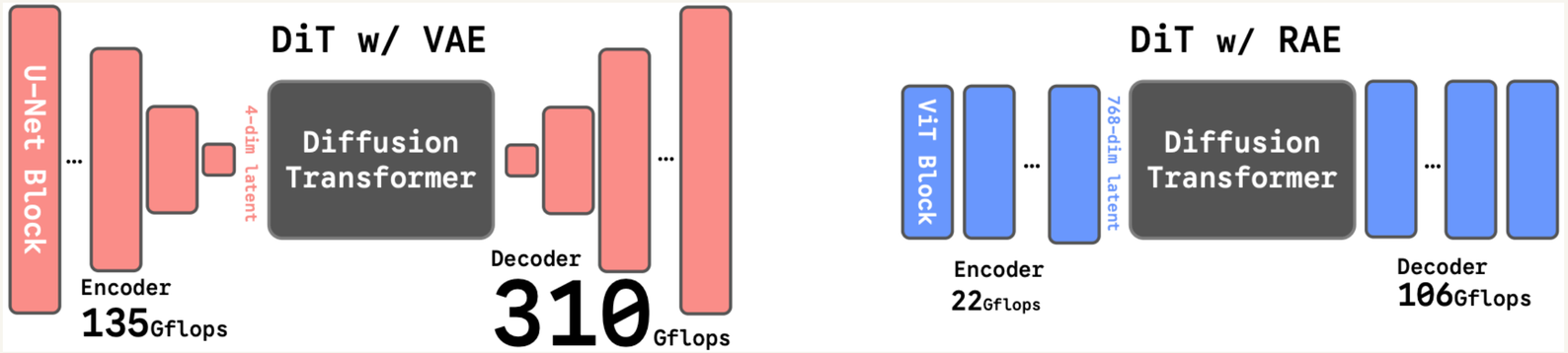
$$\mathcal{L} = \mathbb{E}\left[\|\epsilon_\theta(z_j, j) - \epsilon\|^2\right]$$

Training	Independent generation
Encode an observed image, then corrupt its latent	Start from Gaussian latent noise
Update only the latent prediction network	Repeatedly update the latent, then decode once

$$\hat{y}_0 = D_\psi(\hat{z}_0)$$

A U-Net or a transformer can model this same latent distribution.

Representation autoencoders



RAE paper: replacing a VAE encoder with pretrained representation features.

$$z_0 = E_{\text{pretrained}}(y_0), \qquad \tilde{y}_0 = D_\psi(z_0)$$

Frozen encoder	Use pretrained features, such as DINO, SigLIP, or MAE
Learned decoder	Reconstruct images from those features
Generative prior	Learn the feature distribution with diffusion or flow matching

Zheng et al., Representation Autoencoders (2025)

Generating in representation features



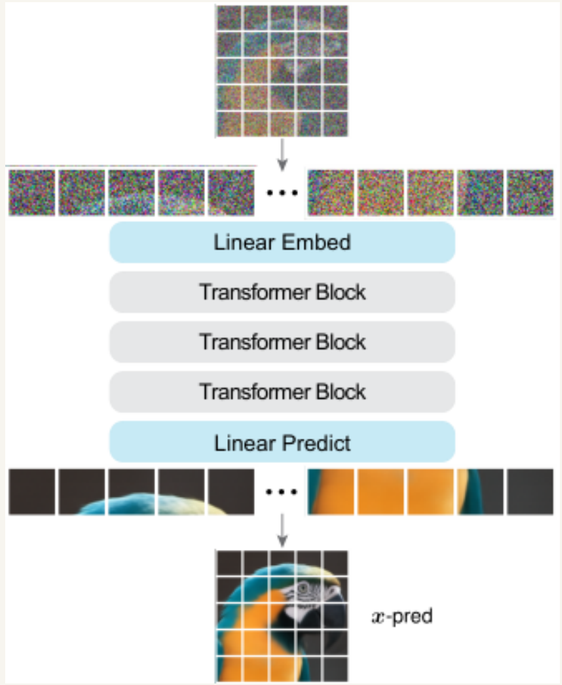
RAE paper: reconstructions from frozen representation features.

The feature space can retain rich information without aggressive channel compression.

The generator must handle its dimensionality and noise scale. The decoder must tolerate imperfect generated features.

Zheng et al., Representation Autoencoders (2025)

JiT: generation directly in pixels



JiT paper: a transformer maps noisy pixel patches to clean pixel predictions.

No pretrained image tokenizer or separate image decoder.

$$\hat{y}_0 = \mathrm{net}_\theta(r_\tau, \tau, c)$$

For the straight noise-to-data path, convert the clean prediction to velocity:

$$v_\theta(r_\tau, \tau, c) = \frac{\hat{y}_0 - r_\tau}{1 - \tau}, \qquad \tau < 1$$

The paper trains this velocity with the flow-matching loss.

Pixel space, compressed latents, and features

Generation space	What the sampler updates	How pixels appear
Pixels, as in JiT	A noisy image tensor	The final clean prediction is already an image
Compressed latents, as in LDM	An autoencoder latent tensor	A trained decoder reconstructs pixels
Representation features, as in RAE	A pretrained encoder feature tensor	A trained feature decoder reconstructs pixels

The sampling method, guidance, and backbone can be chosen separately.

A complete image generator combines these decisions with a conditioning interface.

Training a text-to-image generator

Training examples pair images with descriptions.

$$z_0 = E(y_0), \qquad C = \text{TextEncoder}(\text{description})$$

$$z_\tau = (1-\tau)n + \tau z_0$$

$$\mathcal{L} = \mathbb{E}\left[\|v_\theta(z_\tau, \tau, C) - (z_0 - n)\|^2\right]$$

Image and text encoders supply the training inputs. Condition dropout also trains the null-text mode.

How image features read the text

Image features supply queries. Text features supply keys and values.

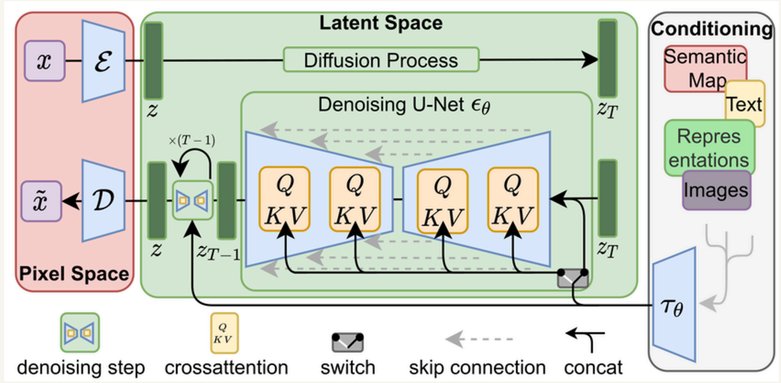
$$Q = UW_Q, \qquad K = CW_K, \qquad V = CW_V$$

$$\mathrm{CrossAttention}(U, C) = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V$$

U: current image features. C: encoded text. d_k: key dimension.

Text remains available while the sampler repeatedly changes the image candidate.

The complete text-to-image pipeline



LDM, Figure 3: an original U-Net-based implementation of the latent text-to-image pipeline.

Initialize	Encode the prompt and draw Gaussian latent noise
Generate	Repeatedly evaluate the conditional network, apply guidance, and update the latent
Decode	Convert the generated clean latent into pixels

Generation needs the prompt and noise. It does not need an image to encode.

Video generation



OpenAI Sora, 2024. Generated video; excerpt from the official release.

How can the same generative framework produce a whole video?

A distribution over a whole video



Frame 1



Frame 2



Frame 3



Frame 4



Frame 5

$$p_\theta(V \mid C), \qquad V = (y^{(1)}, \ldots, y^{(T)})$$

Symbol	Meaning
V	The complete video clip
C	A supplied condition, such as a text prompt or reference image
T	The number of video frames

The model must capture relationships between frames.

Video time and generation time

Across the row: video frame position. Down the table: corruption level.

Corruption level	Frame 1	Frame 2	Frame 3	Frame 4
$j = J$	$y_J^{(1)}$	$y_J^{(2)}$	$y_J^{(3)}$	$y_J^{(4)}$
j	$y_j^{(1)}$	$y_j^{(2)}$	$y_j^{(3)}$	$y_j^{(4)}$
$j = 0$	$y_0^{(1)}$ 	$y_0^{(2)}$ 	$y_0^{(3)}$ 	$y_0^{(4)}$ 

t locates a frame. j locates the whole clip in the corruption process.

One sampling step updates the whole row of video frames.

A diffusion objective for a video

$$V_j = a_j V_0 + b_j \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I)$$

$$\mathcal{L} = \mathbb{E}\left[\|\epsilon_\theta(V_j, j, C) - \epsilon\|^2\right]$$

Quantity	Shape or role
Clean video V_0	$T \times H \times W \times 3$
Noise ϵ	Independent entries with the same tensor shape
Level j	Shared by the jointly corrupted clip

The loss covers the full video tensor. The network models its dependencies.

Ho et al., Video Diffusion Models (2022)

Bidirectional temporal attention

Query frame	Key 1	Key 2	Key 3	Key 4	Key 5
Frame 1	✓	✓	✓	✓	✓
Frame 2	✓	✓	✓	✓	✓
Frame 3	✓	✓	✓	✓	✓
Frame 4	✓	✓	✓	✓	✓
Frame 5	✓	✓	✓	✓	✓

All cells permit access. This diagram shows the attention mask, not learned weights.

At a fixed corruption level, a frame can attend to earlier and later noisy frames.

The network jointly refines a candidate clip.

Frame order in the network



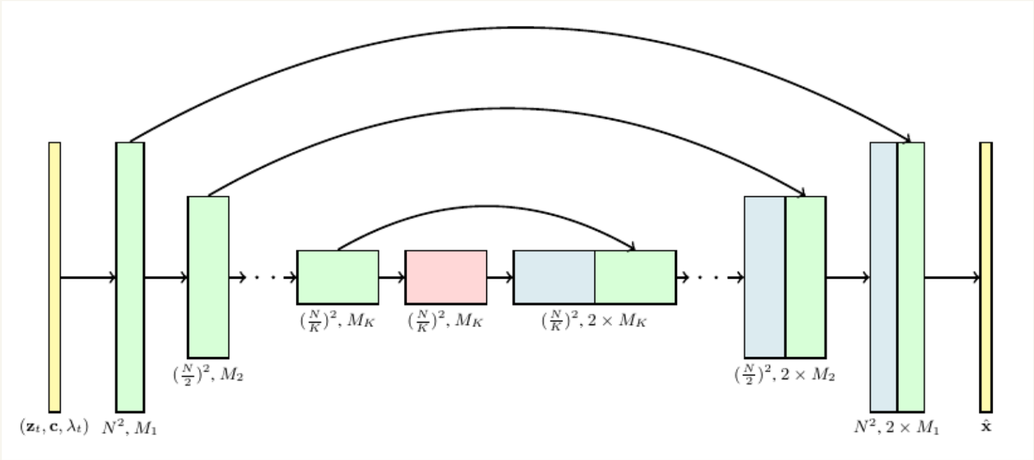
Earlier Middle Later

$$u_{t,h,w} \text{ with position information } (t, h, w)$$

Input	Information it provides
Position in video time	Which frame comes earlier or later
Spatial position	Where a patch lies in a frame
Noise-level embedding	How corrupted the entire candidate is

Bidirectional access still preserves video order.

The original video diffusion network



Video Diffusion Models, Figure 1: a U-Net with spatial and temporal processing.

Spatial computation	Temporal computation
Process structure within a frame	Exchange information across frames
Downsample and upsample spatial features	Preserve the video sequence dimension

Ho et al., Video Diffusion Models (2022)

Spatial and temporal attention

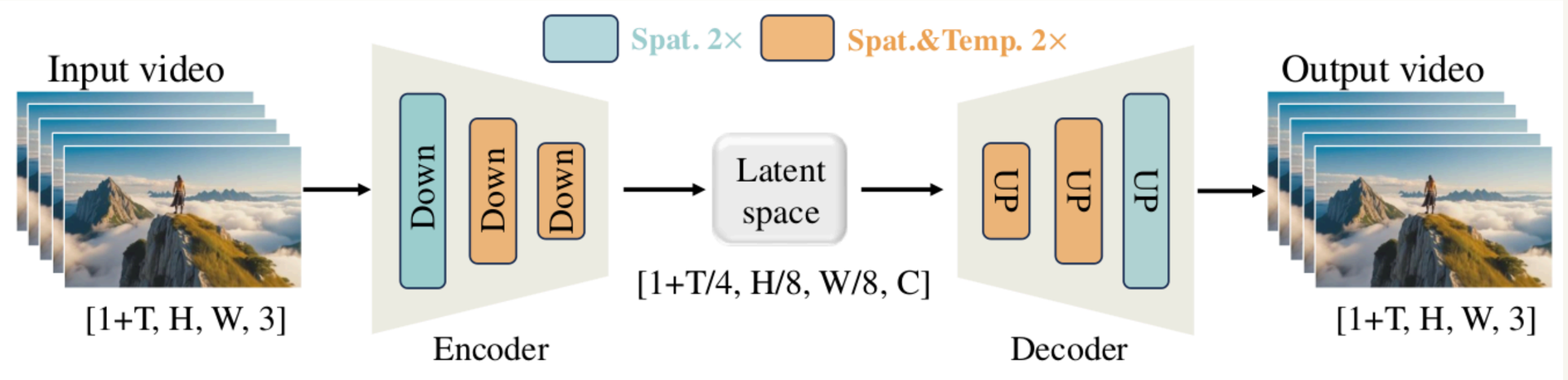
Operation	Tokens that exchange information
Spatial attention	Positions within the same frame
Temporal attention	Positions across frames at a given spatial location

$$\text{SpatialAttention then TemporalAttention}$$

Stacking these operations propagates information across both space and time.

Factorization reduces the cost of a full space-time attention matrix.

A video autoencoder



Wan-VAE, Figure 5: spatial and temporal compression, followed by reconstruction.

$$Z_0 = E_\phi(V_0), \qquad \tilde{V}_0 = D_\psi(Z_0)$$

The latent tensor can compress both spatial resolution and the number of time positions.

A latent time position can represent information from several video frames.

Space-time latent patches

$$Z_j \in \mathbb{R}^{T' \times H' \times W' \times d}$$

$$Z_j \xrightarrow{\text{patchify}} (u_1, \ldots, u_N)$$

$$N = \frac{T'H'W'}{p_t p_h p_w}$$

Symbols	Meaning
T', H', W'	Dimensions of the latent video grid
d	Latent channels
p_t, p_h, p_w	Patch sizes in latent time and space

The cost of space-time attention

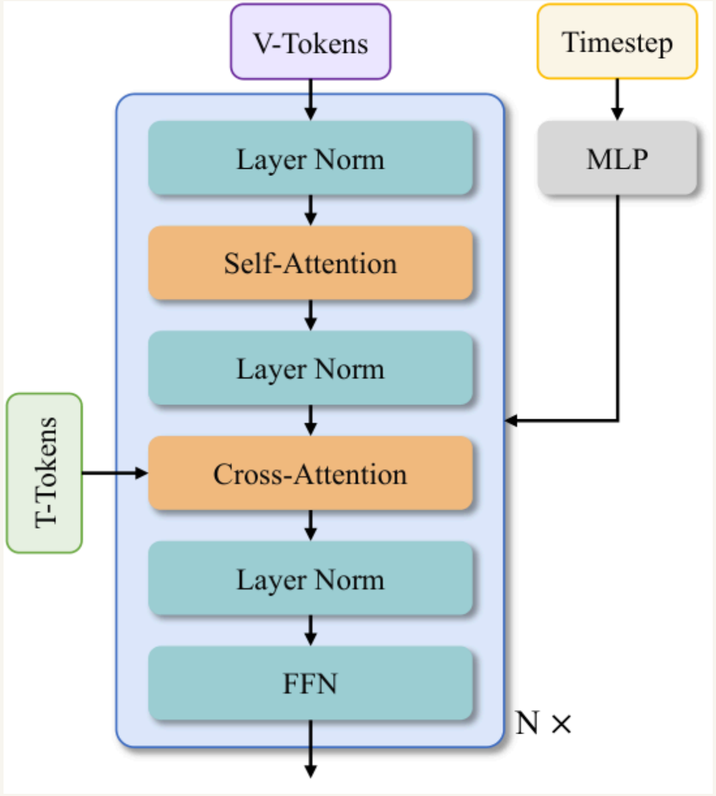
$$N = T'S, \qquad S = \text{spatial tokens per latent frame}$$

Architecture	Approximate attention work
Full space-time attention	$O(T'^2 S^2)$
Factorized spatial and temporal attention	$O(T' S^2 + S T'^2)$

More frames and higher resolution increase token count and repeated sampling cost.

Compression and attention design determine which clip sizes are practical.

A video diffusion transformer block



Wan, Figure 10: the video transformer block.

Video self-attention

Connect noisy space-time tokens

Text cross-attention

Read the supplied prompt representation

Noise conditioning

Modulate the block using the generation coordinate

Training a joint video flow model

$$Z_0 = E_\phi(V_0), \qquad Z_\tau = (1-\tau)\eta + \tau Z_0$$

$$\mathcal{L} = \mathbb{E}\left[\left\| v_\theta(Z_\tau, \tau, C) - (Z_0 - \eta)\right\|^2\right]$$

η: Gaussian noise with the shape of the complete video latent.

Encode	A training video and its supplied condition
Corrupt	Mix the entire clean latent with noise
Predict	Regress the velocity tensor for the entire clip

The video VAE stays fixed while this objective updates the transformer.

Sampling an entire latent clip

$$Z_{\tau=0} \sim \mathcal{N}(0, I)$$

$$Z_{\tau+\Delta\tau} \approx Z_\tau + \Delta\tau\, v_\theta(Z_\tau, \tau, C)$$

$$\hat{V} = D_\psi(Z_{\tau=1})$$

Each integration step	Evaluate the current complete clip
Update	All latent time positions together
At the end	Decode the generated latent into video frames

Text-to-video generation

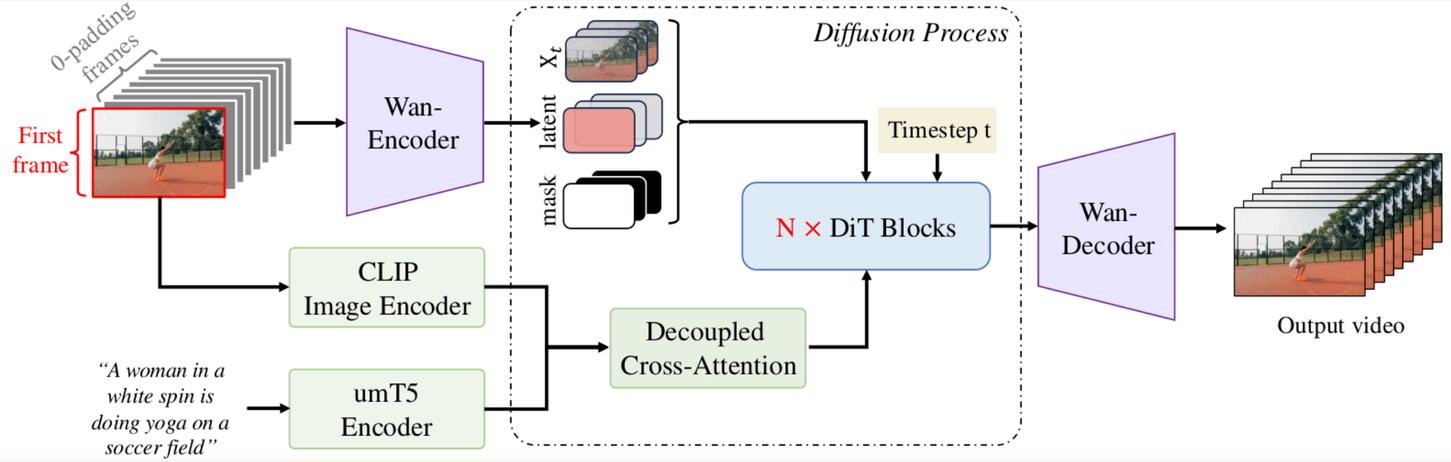
$$C = \text{TextEncoder}(\text{prompt})$$

$$p_\theta(V \mid C)$$

Text input	What it can specify
Scene and appearance	Subjects, objects, environment, and style
Events and motion	Descriptions of what should happen
Camera information	Viewpoint and intended camera movement

Text conditions a learned distribution; it does not certify the dynamics.

Image-to-video generation



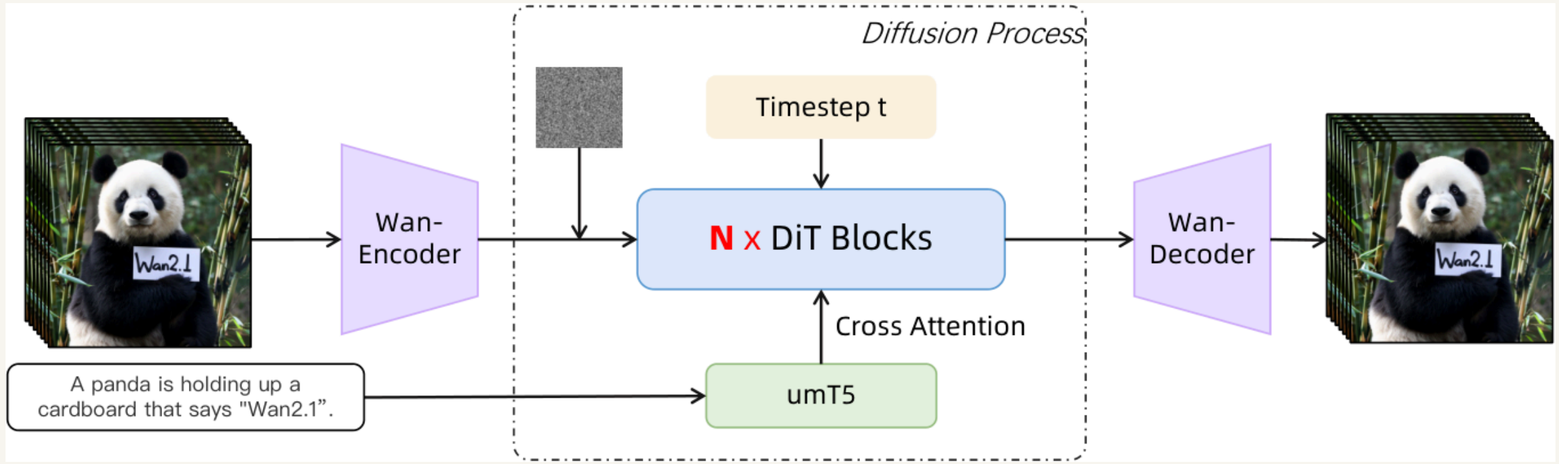
Wan, Figure 18: reference-image features and a masked visual condition enter the generator.

Available condition	Generated object
A reference image and optional text	A complete candidate video
Image features and a visual-condition mask	Jointly refined video latents

A supplied initial image allows joint generation of the remaining frames.

Wan Team, Wan (2025)

The complete Wan pipeline



Wan, Figure 9: latent video, prompt encoding, video transformer, and decoder.

During generator training	During sampling
Encode observed videos; regress known targets	Initialize latent noise; evaluate the transformer repeatedly
Freeze the VAE and text encoder	Decode the generated latent at the end

Wan Team, Wan (2025)

Evaluating generated videos

Question	What to inspect
Do frames look plausible?	Visual detail and artifacts
Does the clip form a coherent event?	Identity, motion, and persistence
Does it follow the condition?	Prompt and reference-image adherence
Do the dynamics make sense?	Contacts, occlusion, and object-state changes

One perceptual score cannot answer every question about the video.

Ho et al., Video Diffusion Models (2022) · Wan Team, Wan (2025) · OpenAI, Sora technical report (2024)

The joint video generation recipe

Component	Role
Video encoder	Represent observed clips in a latent space
Space-time transformer	Predict updates using the complete noisy candidate
Conditioning	Supply text or observed visual evidence
Diffusion or flow sampler	Refine the complete latent clip
Video decoder	Recover pixel frames from the final latent

Next: how can this generator predict futures conditioned on experience and interaction?