UC San Diego

Scalable Diffusion Models with Transformers

William Peebles, Saining Xie · UC Berkeley & New York University · 2023

Presentation by Purush Ram
UCSD Electrical & Computer Engineering Department | Spring ‘26

The Diffusion Roadmap

Foundational Concept → Real Model → Make it better → Diffusion Transformer

01. Diffusion Math

Define forward and reverse markov chains for denoising.

Sohl-Dickstein, 2015

02. DDPM

Turn the reversed chain into effective, efficient noise prediction.

Ho et.al, 2020

03. U-Net

The backbone architecture of Diffusion models.

Ronneberger et.al, 2015

04. Latent Diffusion

Running denoising in a compressed VAE latent space.

Rombach et.al, 2022

05. DiT

No more U-Net! Replace it with a Transformer.

Peebles et.al, 2023

The Mechanics of Diffusion

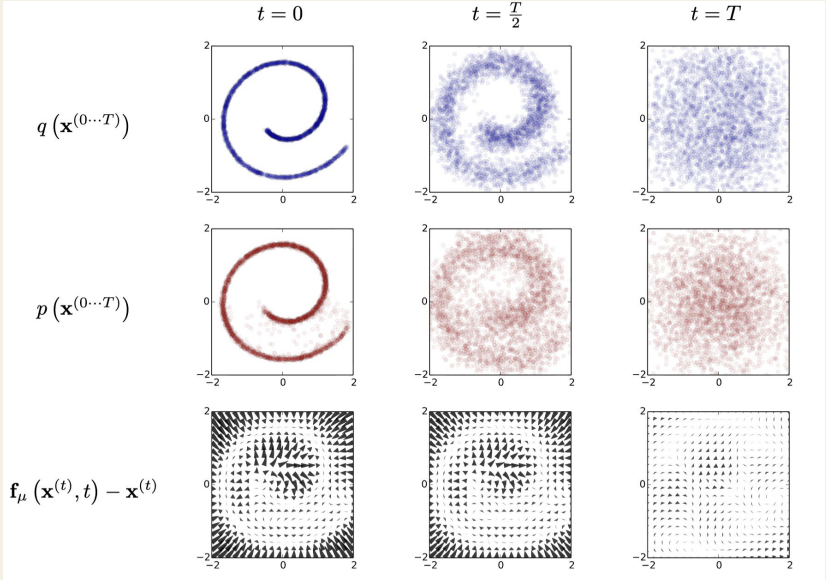
GOAL: Maximize log-probability of clean image (x_0)

- **q (Forward):** A fixed rule destroying data by adding Gaussian noise step by step.
- **p (Reverse):** A neural network (U-Net, for instance) that learns to predict and remove that noise

Intractable Problem

$$p(x_0) = \int p_\theta(x_{0:T})\, dx_{1:T}$$

Calculating **every infinite path** of noise is impossible for a computer.



The Mechanics of Diffusion

Intractable Integrals to Variational Lower Bounds

1. The Intractable Integral

We must marginalize over all possible noise paths to find the data probability.

$$\log p_\theta(x_0) = \log \int p_\theta(x_{0:T}) dx_{1:T}$$

This integral is impossible to do! Essentially is a one-shot

2. Forward Injection

Break it into denoising steps:

Multiply by q/q to transform the integral into a manageable expectation.

$$\log p_\theta(x_0) = \log \mathbb{E}_q \left[\frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)} \right]$$

3. Jensen's Inequality

Swap log and expectation to establish a computable Lower Bound floor.

$$\log p_\theta(x_0) \geq \mathbb{E}_q \left[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)} \right]$$

Maximizing this bound (VLB) is the core of diffusion training.

This is just a few steps of extremely long proofs done by Sohl-Dickstein.

Problem: These strict, rigorous mathematics are not visually scalable!

Denoising Diffusion Probabilistic Models (DDPM)

Sohl-Dickstein's Complexity (Strict VLB)

Issue: Tries to predict the exact mean (μ) of the reverse distribution.

The strict proofs say each timestep's loss must be multiplied by complex variance coefficients.

$$L_{t-1} = \mathbb{E}_q[\frac{1}{2\sigma_t^2}\|\tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t)\|^2]$$

These strict weights prioritize steps closer to clean image (t ≈ 0)

Model obsesses over small pixel imperfections and loses global structure.

VS

DDPM's Simplified Objective

Solution: Predicts the raw noise matrix (ε) instead of mean.

Every timestep is treated with equal importance (weight = 1).

$$L_{\text{simple}} = \mathbb{E}_{t,x_0,\epsilon}[\|\epsilon - \epsilon_\theta(x_t, t)\|^2]$$

Result: This forces the network to learn macro-structures first, which results in much higher accuracy image gen.

Denoising Diffusion Probabilistic Models

DDPM (Ho, et. al) collapses training to one sampled timestep.

- Direct noising: $x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{(1-\bar{\alpha}_t)}\,\varepsilon$
- Noise loss: $L_{simple} = E_{t,x_0,\varepsilon} \| \varepsilon - \varepsilon_\theta(x_t,t) \|^2$
- Key abstraction: ε_θ is a denoising network slot.

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

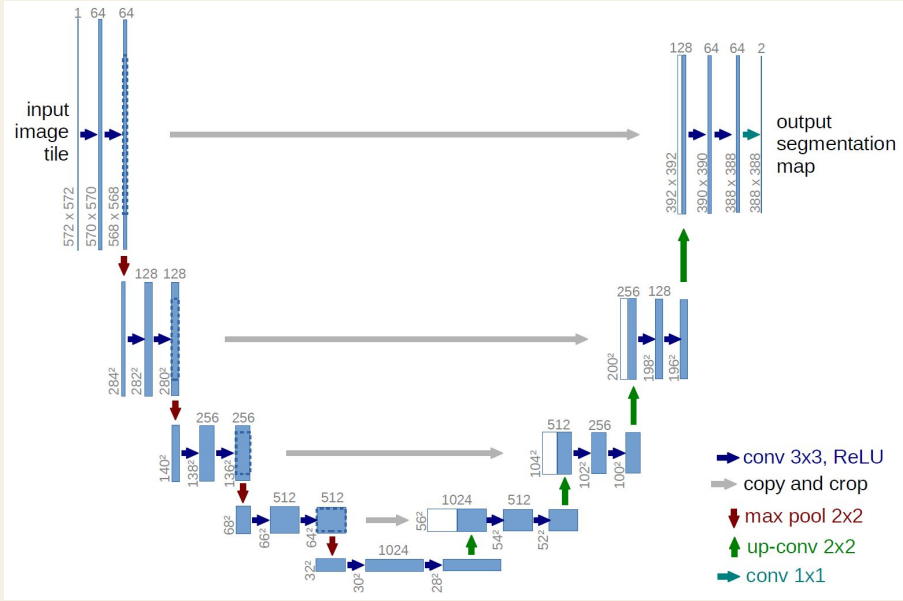
Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Enabling Global Context: The U-Net

U-Net became the underlying architecture for image denoising

- Why it works: local convolutions + downsampled context + skip connections.
- Where it sits: x_t, t, c → ε_θ → predicted noise.
- Why DiT can challenge it: the objective only asks for a denoiser.



Pixel Space Was Too Expensive

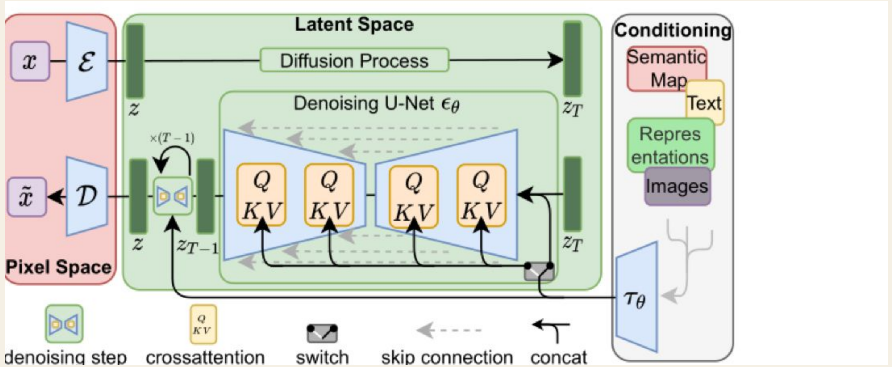
The repeated reverse loop makes representation size a first-order cost.

- Pixel state: 512×512×3 = 786K scalar values.
- Latent state: 64×64×4 = 16K values for f=8.
- Shrink the state once; save work at every denoising step.

Stable Diffusion = Latent Diffusion

Keep DDPM noise prediction, but denoise VAE latents instead of pixels.

- Encode: $x \to z = E(x)$; decode: $\hat{x} = D(z)$.
- Train: $L_{LDM} = E \, || \, \varepsilon - \varepsilon_\theta(z_t, t, c) \, ||^2$
- The U-Net works on latent structure instead of raw pixels, much smaller dimensions



The Handoff To DiT

LDM answers where to denoise; DiT asks what network should denoise.

- Hold fixed: frozen VAE, latent diffusion objective, conditioning setup.
- Swap: U-Net $\varepsilon_\theta(z_t,t,c)$ → Transformer $\varepsilon_\theta(z_t,t,c)$.
- This makes DiT a controlled architecture-scaling experiment.

DiT's Controlled Experiment

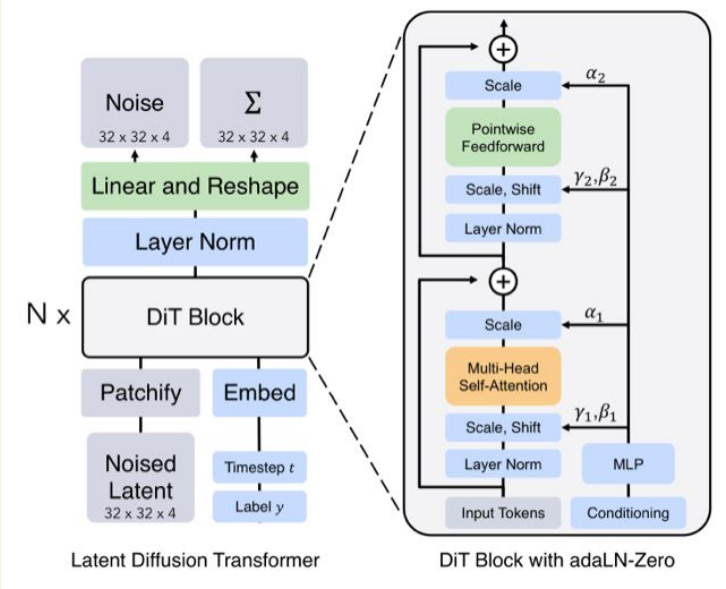
Was the latent U-Net essential, or just inherited?

- Input stays a noisy latent z_t with timestep t and class c.
- Backbone changes from convolutional U-Net blocks to ViT-style blocks.
- Question: does denoiser compute scale quality predictably?

DiT Converts A Latent Grid Into Tokens

Patchify, run Transformer blocks, then unpatchify predictions.

- Patch embedding maps each p×p latent patch to one token.
- Tokens carry position plus conditioning from t and c.
- The final layer predicts patchwise noise/covariance, then reshapes to the latent grid.



The Loss Is Still Diffusion

The paper isolates architecture by keeping the denoising target fixed.

- $L_{DiT} = E_{z_0,t,\varepsilon,c} || \varepsilon - \varepsilon_\theta(z_t,t,c) ||^2$
- Same target: the added noise ε in latent space.
- Different hypothesis class: ε_θ is a Transformer instead of a U-Net.

Patch Size Is The Compute Knob

p sets token count; token count sets attention cost.

- $T = (I / p)^2$ tokens for an I×I latent grid.
- For I=32: p=8 → 16 tokens; p=4 → 64; p=2 → 256.
- Smaller p raises Gflops with almost the same parameter count.

Conditioning Is The Hard Part

Every block needs to know both the noise level and the class.

- t tells the block how corrupted z_t is.
- c tells the denoising process what sample it should become.
- DiT tests four mechanisms: in-context tokens, cross-attention, adaLN, adaLN-Zero.

adaLN-Zero

Conditioning starts as identity, then learns residual denoising.

- A conditioning MLP maps (t,c) → γ, β, α for each block.
- γ,β modulate LayerNorm; α gates the residual branch.
- Initialize α=0, so deep DiT blocks begin as near-identity maps.

Scaling Axes

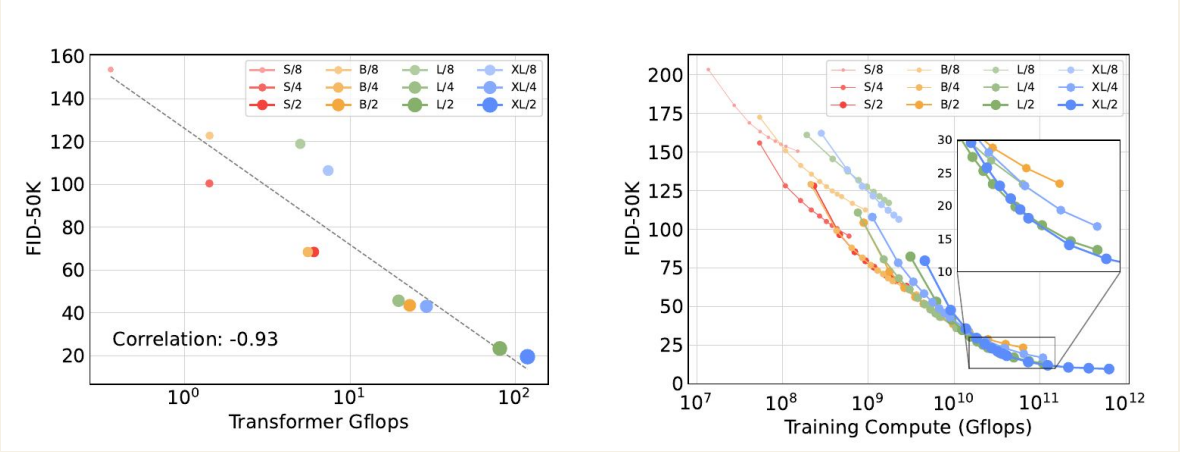
DiT makes compute an experimental variable, not a vague scale claim.

- Model size: S/B/L/XL changes depth, width, and heads.
- Patch size: p=8/4/2 changes token count $T=(I/p)^2$.
- Together: 12 models test whether Gflops predict FID across design choices.

Compute Predicts FID

Across the DiT grid, more Transformer Gflops means better samples.

- Reported correlation: r ≈ −0.93 between forward-pass Gflops and FID.
- Lower FID is better; the trend holds across model size and patch size.
- Interpretation: quality improves when compute is spent inside ε_θ.



Why Gflops, Not Parameters

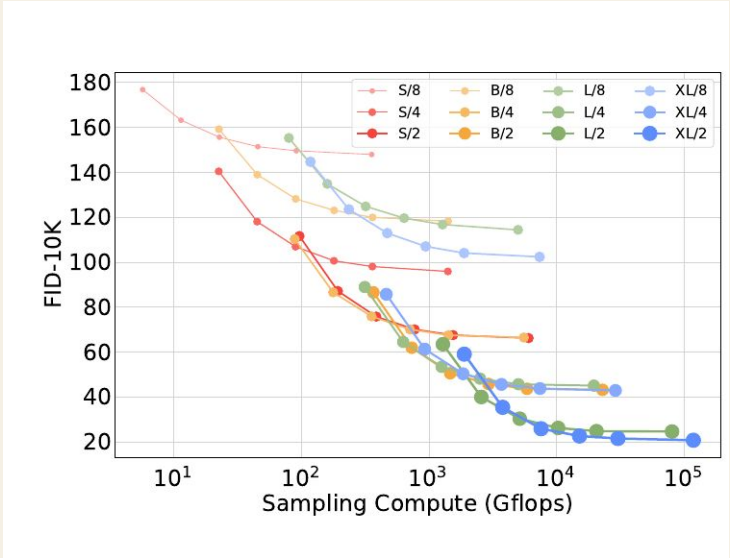
Patch size changes compute while leaving parameter count nearly fixed.

- Same DiT weights, smaller p → more tokens → more attention/MLP work.
- Different models can land at similar Gflops and similar FID.
- The paper’s scaling variable is per-step denoiser compute.

Sampling More Is Not Enough

Extra reverse steps do not fully compensate for a weak denoiser.

- Two knobs: K sampling steps versus Gflops per denoising step.
- Figure 10: small models plateau even as K increases.
- Better samples come primarily from a stronger per-step model.



Headline Benchmark

The scaling story pays off on ImageNet 256.

- DiT-XL/2 with classifier-free guidance: FID 2.27.
- Compare: LDM-4-G 3.60; ADM-G 3.94.
- The benchmark validates the mechanism, not a new diffusion loss.

Class-Conditional ImageNet 256×256					
Model	FID↓	sFID↓	IS↑	Precision↑	Recall↑
BigGAN-deep [2]	6.95	7.36	171.4	0.87	0.28
StyleGAN-XL [53]	2.30	4.02	265.12	0.78	0.53
ADM [9]	10.94	6.02	100.98	0.69	0.63
ADM-U	7.49	5.13	127.49	0.72	0.63
ADM-G	4.59	5.25	186.70	0.82	0.52
ADM-G, ADM-U	3.94	6.14	215.84	0.83	0.53
CDM [20]	4.88	-	158.71	-	-
LDM-8 [48]	15.51	-	79.03	0.65	0.63
LDM-8-G	7.76	-	209.52	0.84	0.35
LDM-4	10.56	-	103.49	0.71	0.62
LDM-4-G (cfg=1.25)	3.95	-	178.22	0.81	0.55
LDM-4-G (cfg=1.50)	3.60	-	247.67	**0.87**	0.48
DiT-XL/2	9.62	6.85	121.50	0.67	**0.67**
DiT-XL/2-G (cfg=1.25)	3.22	5.28	201.77	0.76	0.62
DiT-XL/2-G (cfg=1.50)	**2.27**	**4.60**	**278.24**	0.83	0.57

Systems Interpretation

Replacing U-Net with Transformer changes the optimization surface.

- Compute shifts toward attention, MLPs, normalization, and dense matmuls.
- That opens mature systems tools: FlashAttention, quantization, compiler fusion, parallelism.
- Caveat: these are implications; DiT does not benchmark them.

Limitations

The paper proves a scaling direction, not every diffusion claim.

- Scope: class-conditional ImageNet, not broad text-to-image/video evaluation.
- Baseline: no fully compute-matched scaled U-Net comparison.
- Cost: dense attention is $O(T^2)$, so small patches and video tokens get expensive.

Why This Matters Beyond Images

The denoiser can act on any structured continuous variable.

- Images: denoise a latent token grid toward a sample.
- Robotics: denoise an action sequence/chunk toward commands.
- So DiT’s question becomes: what backbone should a diffusion policy denoise with?

Robot Control

Architecture scaling has a hardware price.

- Autoregressive VLA: token-by-token decode, KV/cache traffic, memory pressure.
- Diffusion policy: K repeated dense denoising passes, often compute-heavy.
- Project question: how do these workloads land on an RTX 4060 roofline?

Takeaways And Discussion

DDPM defines the slot; LDM moves it; DiT makes it scale.

- DDPM turns reverse diffusion into ε-prediction.
- LDM keeps the loss but moves denoising into latent space.
- DiT swaps the denoiser for a Transformer and finds compute predicts quality.
- Discussion: fixed compute → larger denoiser, fewer steps, or better perception?