Diffusion Transformers (DiT)

Scalable Diffusion Models with Transformers

Camille L'Herminé

December 10, 2025

Overview: An Architectural Shift

The Status Quo: U-Net

- Historically, diffusion models relied on the **U-Net** backbone (e.g., Stable Diffusion).
- **Problem:** U-Nets have complex inductive biases that make them difficult to scale effectively.

The Inspiration: Transformers

- Transformers (e.g., GPT, ViT) have shown massive success due to scalability.
- **Idea:** Can we replace the U-Net with a standard Transformer?

The Core Proposal

Replacing the U-Net backbone with a **Diffusion Transformer (DiT)** to inherit the scaling properties of the Transformer architecture, inspired by Vision Transformers (ViT).

Methodology: How DiT Works

Unlike standard ViT (which patches raw pixels), DiT operates in **Latent Space**.

The Processing Pipeline

$$\text{Image} \xrightarrow{\text{VAE Encoder}} \text{Latent Space} \xrightarrow{\text{Patching}} \text{Transformer Input}$$

- **1. Compression:** A Variational Autoencoder (VAE) compresses the image into a lower-dimensional representation (Latent Space).
- **2. Patching:** The latent feature map is divided into a sequence of patches (tokens).
- **3. Processing:** These tokens are fed into the Transformer blocks (DiT), similar to how words are processed in LLMs.

Performance Assessment

The authors analyze the model through the lens of **Forward pass Complexity** (Gflops).

- **Scaling Metric:** Performance is measured against total Gflops (computational cost per forward pass, not nb of parameters as they don't account for image resolution for example).
- **Quality Metric:** Fréchet Inception Distance (FID) - *lower is better*.

Key Findings

1. **Scalability:** Increasing Transformer depth/width or input tokens consistently lowers FID.
2. **SOTA Results:** The largest model (DiT-XL/2) outperforms all prior diffusion models on ImageNet.

Evaluation Metric: Fréchet Inception Distance (FID)

Goal: Measure the quality and diversity of generated images. (*Lower score is better.*)

1. The Mechanics: Feature Extraction

1. **Extractor:** Both real and generated images are fed through the **Inception** – **v3** CNN (weights are frozen).
2. **Features:** The activations from a deep intermediate layer (2048-dim vector) are used as the image's abstract representation.
3. **Distributions:** The feature sets are modeled as two separate **multivariate Gaussian distributions** (P_r for real, P_g for generated).

2. The Calculation: Fréchet Distance

The FID score measures the distance between the two Gaussian distributions, P_r and P_g, characterized by means ($\boldsymbol{\mu}$) and covariance matrices ($\boldsymbol{\Sigma}$):

$$\text{FID} = ||\boldsymbol{\mu}_r - \boldsymbol{\mu}_g||^2 + \text{Tr}\left(\boldsymbol{\Sigma}_r + \boldsymbol{\Sigma}_g - 2(\boldsymbol{\Sigma}_r\boldsymbol{\Sigma}_g)^{1/2}\right)$$

- **Mean Term ($||\boldsymbol{\mu}||^2$):** Measures the **Realism**/**Quality** (how close the generated image centers are to the real center).
- **Covariance Term ($\text{Tr}(\boldsymbol{\Sigma})$):** Measures the **Diversity**/**Mode Coverage** (how well the generated images cover the feature space).

Context DDPM: Forward & Reverse Trajectories

We model the data x_0 via a sequence of latent variables $x_1, \ldots, x_T$.

1. The Forward Process (q - Fixed): A Markov chain that gradually adds Gaussian noise according to schedule β_t.

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}), \quad q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t \mathbf{I})$$

2. The Reverse Process (p_θ - Learned): A Markov chain starting from $p(x_T) = \mathcal{N}(0, \mathbf{I})$ to recover data.

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$$

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

Algorithm 1: Training (The Learning)

The objective is to minimize the difference between true noise ϵ and predicted noise ϵ_θ.

Algorithm 1 Training the Diffusion Model

Sample data: $x_0 \sim q(x_0)$ **Sample** timestep: $t \sim \text{Uniform}(\{1, \dots, T\})$ **Sample** noise: $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ **Compute** loss (Gradient Descent):

$$\nabla_\theta \|\epsilon - \epsilon_\theta(\underbrace{\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon}_{\text{Noisy input } x_t}, t)\|^2$$

converged

Once ϵ_θ is trained, we reverse the process.

Algorithm 2 Sampling (Generation)

Start $x_T \sim \mathcal{N}(0, \mathbf{I})$ $t = T, \dots, 1$ Sample $z \sim \mathcal{N}(0, \mathbf{I})$ if $t > 1$, else $z = 0$ **Update** x_{t-1}:

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right) + \sigma_t z$$

Return x_0

Advanced Training: Learning the Variance

Standard DDPM: Fixed variance $\Sigma_t = \beta_t \mathbf{I}$.
Improved DDPM (Nichol & Dhariwal): We learn Σ_θ to improve log-likelihood and sampling speed.

The Challenge: L_{simple} (MSE) works best for image quality but cannot learn variance (no gradient signal). L_{VLB} learns variance but is unstable for the mean.

Hybrid Loss Strategy

We use a combined objective with a stop-gradient operator:

$$L_{\text{hybrid}} = L_{\text{simple}}(\mu_\theta) + \lambda L_{\text{VLB}}(\Sigma_\theta)$$

- **Mean (μ_θ):** Trained via L_{simple} (Predicting noise ϵ).
- **Variance (Σ_θ):** Trained via L_{VLB} (Minimizing KL).
- **Result:** Faster sampling (fewer steps needed) and higher quality.

Conditional Generation: Classifier-Free Guidance

Goal: Generate samples conditioned on class c (e.g., "Golden Retriever").
The Insight (Bayes Rule): The direction to "more class correctness" is the difference between the *conditional* and *unconditional* score.

$$\nabla_x \log p(c|x) \propto \nabla_x \log p(x|c) - \nabla_x \log p(x)$$

The Formula (CFG): Instead of a separate classifier, we mix predictions from a single model:

The Guidance Equation

$$\hat{\epsilon}_\theta(x_t, c) = \underbrace{\epsilon_\theta(x_t, \emptyset)}_{\text{Uncond.}} + \underbrace{s}_{\text{Scale}} \cdot (\underbrace{\epsilon_\theta(x_t, c)}_{\text{Cond.}} - \epsilon_\theta(x_t, \emptyset))$$

- **Training:** Randomly replace label c with null $\emptyset$ (e.g., 10% dropout).
- **Scale $s > 1$:** Pushes the image away from "generic" and towards "specific."

Input Specifications: The "Patchify" Layer

Input: Latent representation z (not pixels).
Example: Image $256^2 \times 3 \rightarrow$ Latent $32^2 \times 4$.

The Process:

1. "Patchify" input $I \times I$ into patches of size $p \times p$.
2. Flatten into sequence of length $T = (I/p)^2$.
3. Add Positional Embeddings (Sine-Cosine).

Gflops vs. Patch Size (p):

- Halving $p \rightarrow$ Quadruples T.
- Significantly increases Gflops (Compute).
- Parameter count remains constant.

Conditioning Strategies: Designing the DiT Block

The Challenge: How do we inject Time (t) and Class (c) into the Transformer?

Variant	Mechanism	Outcome
In-Context	Append t, c as extra tokens (like [CLS]).	Low Cost, Low Perf.
Cross-Attn	Add extra attention layer for t, c.	High Cost (+15% Gflops).
adaLN	Regress Layer Norm params (γ, β) from t, c.	Efficient.
adaLN-Zero	**adaLN + Zero-Initialized Gating.**	**Best FID**

Why adaLN-Zero Wins:

- **Efficiency:** Modifies existing layers rather than adding new ones.
- **Identity Initialization:** The α gate starts at 0, making the block an Identity function initially. This stabilizes deep training (similar to ResNet/U-Net tricks).

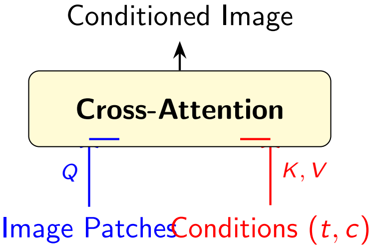
Conditioning via Cross-Attention

Method: Keep Image tokens and Condition tokens separate.
The Image tokens "look at" the Condition tokens using a dedicated attention layer.

The Mechanism (Q, K, V): In standard Self-Attention, an image looks at itself. In Cross-Attention:

- **Query (Q):** Comes from Image Tokens.
- **Key (K):** Comes from Condition (t, c).
- **Value (V):** Comes from Condition (t, c).

"Every image patch asks the Class and Time embeddings for relevant info."



Calculates: Softmax(QK^T)V

Adaptive Layer Normalization (adaLN)

Goal: Dynamically predict the Layer Norm parameters (γ, β) based on conditions $(\mathbf{t}, \mathbf{c})$.

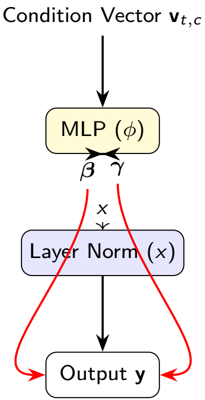
1. **The Prediction Network (The MLP)**
   - **Fusion:** Timestep t and Class c embeddings are summed to create a single Condition Vector.

     $$\mathbf{v}_{t,c} = \mathbf{t}_{\text{emb}} + \mathbf{c}_{\text{emb}}$$

   - **Regression:** This vector is fed into a small, separate MLP (ϕ).
   - **Output:** The MLP projects the result into $2d$ dimensions, which are split to form the adaptive parameters:

     $$(\boldsymbol{\gamma}, \boldsymbol{\beta}) = f_{\text{MLP}}(\mathbf{v}_{t,c})$$

     (The weights of the MLP are what the model actually learns.)



2. The Modulation (The Application)

The predicted $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ replace the static parameters in the Layer Norm formula:

$$\mathbf{y} = \quad \boldsymbol{\gamma}(\mathbf{t}, \mathbf{c}) \quad \odot \text{LayerNorm}(\mathbf{x}) + \quad \boldsymbol{\beta}(\mathbf{t}, \mathbf{c})$$

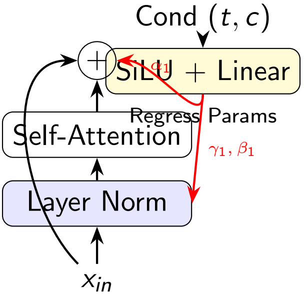
The adaLN-Zero Block

Mechanism: Instead of learning static Scale/Shift params, we predict them from the condition (c, t) using an MLP.

The Regression:

$$(t, c) \xrightarrow{\text{MLP}} (\gamma, \beta, \alpha)$$

1. γ, β**:** Modulate the normalization (Scale & Shift).
2. α **(Gate):** Scales the residual output.



Zero-Initialization: We initialize $\alpha = 0$.

$$x_{out} = x_{in} + \underbrace{\alpha}_{\approx 0} \cdot \text{Block}(x_{in})$$

Adaptive Normalization: The Dynamic Gate

Goal: Replace static parameters with dynamic predictions from conditions $(\mathbf{t}, \mathbf{c})$.

1. The Static Baseline (Standard LN)

$$\text{LN}(\mathbf{x}) = \frac{\mathbf{x} - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

$$\mathbf{y} = \underbrace{\gamma}_{\text{Static Scale}} \odot \text{LN}(\mathbf{x}) + \underbrace{\boldsymbol{\beta}}_{\text{Static Shift}}$$

$\gamma, \boldsymbol{\beta}$ are fixed vectors; they do not change for a "cat" vs. a "car."

2. The Dynamic Control (AdaLN)

$$(\gamma, \boldsymbol{\beta}) = f_{\text{MLP}}(\mathbf{t}, \mathbf{c})$$

$$\mathbf{y} = \underbrace{\gamma(\mathbf{t}, \mathbf{c})}_{\text{Dynamic Scale}} \odot \text{LN}(\mathbf{x}) + \underbrace{\boldsymbol{\beta}(\mathbf{t}, \mathbf{c})}_{\text{Dynamic Shift}}$$

The ***MLP*** *is the learned function f that predicts the required γ and $\boldsymbol{\beta}$ for every condition.*

Key Conclusion: Feature Gating

- The **MLP** is trained to map the condition vector to the normalization parameters.
- This modulation allows the network to **reconfigure its features**: amplifying features (e.g., fur for a cat prompt) and suppressing others for a specific generation task.

Final Decoder and DiT Design Space

Goal: Convert the latent token sequence into spatial predictions $(\epsilon_\theta, \Sigma_\theta)$ for the diffusion step.

Decoder Process: Sequence $\rightarrow$ Spatial Output

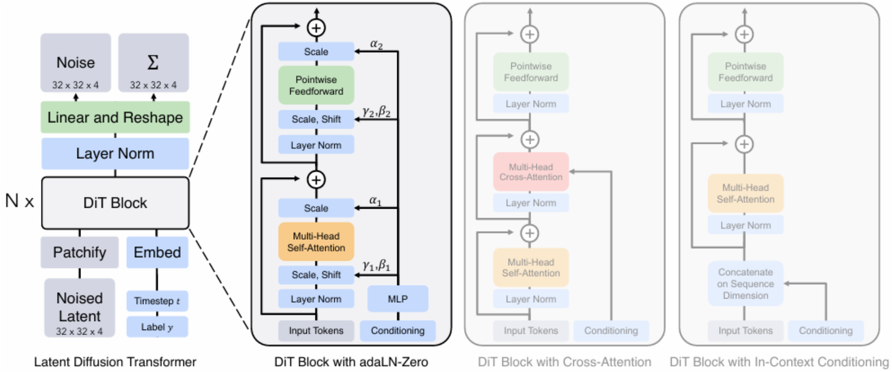
1. **Layer Norm:** Apply the final, adaptive Layer Norm to the token sequence.
2. **Linear Projection:** Each token of hidden dimension d is linearly decoded (projected) back into the full patch content.
3. **Output Shape:** The projection must be $p \times p \times$ **2C** (C for noise ϵ, C for covariance Σ).
4. **Rearrange:** The long sequence of projected patches is reshaped back into the $I \times I \times 2C$ spatial latent grid.

This reverses the "Patchify" step.

The DiT Design Space

The authors explored three major architectural axes to find the optimal DiT model:

1. **Patch Size (p)**: Controls the sequence length T and the total **Gflops** (compute cost).
2. **Transformer Block Architecture**: Controls the method of **Conditioning** (t, c). (*The best was adaLN-Zero*).
3. **Model Size**: Controls the **Depth** (number of blocks) and **Width** (hidden dimension d) of the Transformer.



Noise
32 x 32 x 4
Σ
32 x 32 x 4
Linear and Reshape
Layer Norm
N x
DiT Block
Patchify
Embed
Noised Latent
32 x 32 x 4
Timestep t
Label y
Latent Diffusion Transformer
Scale
α_2
Pointwise Feedforward
Scale, Shift
γ_2, β_2
Layer Norm
Scale
α_1
Multi-Head Self-Attention
Scale, Shift
γ_1, β_1
Layer Norm
MLP
Input Tokens
Conditioning
DiT Block with adaLN-Zero
Pointwise Feedforward
Layer Norm
Multi-Head Cross-Attention
Layer Norm
Multi-Head Self-Attention
Layer Norm
Input Tokens
Conditioning
DiT Block with Cross-Attention
Pointwise Feedforward
Layer Norm
Multi-Head Self-Attention
Layer Norm
Concatenate on Sequence Dimension
Input Tokens
Conditioning
DiT Block with In-Context Conditioning

Training and Evaluation Strategy

Core Principle: Hyperparameters retained from ADM; focus on architectural scaling.

1. **Training Environment**
   - **Data:** ImageNet (Class-Conditional).
   - **Model Naming:** DiT-XL/2 $\rightarrow$ XLarge config, patch size $p = 2$.
   - **Hyperparameters:** Retained from ADM. Constant Learning Rate $\mathbf{1 \times 10^{-4}}$, Batch Size 256.
   - **Initialization:** Final linear layer initialized with zeros (adaLN-Zero philosophy).
   - **Stability:** Training was highly stable across all configs, requiring **no** LR warmup or regularization.
   - **Final Model:** Exponential Moving Average (EMA, decay 0.9999).

2. **Diffusion and Evaluation**
   - **VAE:** Off-the-shelf encoder/decoder from Stable Diffusion (Downsample $\times 8$).
     $\mathbf{256^2} \rightarrow \mathbf{32^2}$ Latent Space.
   - **Diffusion Params:** Retained from ADM ($t_{\max} = 1000$, linear schedule, learned $\mathbf{\Sigma}_\theta$).
   - **Primary Metric:** Fréchet Inception Distance (FID-50K). (Lower is better).
   - **Key Finding (Scaling):** Scaling the Transformer backbone (larger model size or smaller patch size p) consistently **improves FID** at all stages of training.

Experimental Result: Conditioning Strategy

Question: Which method injects $(\mathbf{t}, \mathbf{c})$ most effectively?

Key Finding (Figure 5): The DiT block architecture critically affects final image quality.

- Winner: **adaLN-Zero** yields the lowest **FID** (best quality).
- **Efficiency:** adaLN-Zero achieves this while being the most $\mathbf{compute-efficient}$ (lowest Gflops).
- **Quality Impact:** At 400K steps, adaLN-Zero achieves an FID nearly **half** that of the 'In-Context' model, proving the mechanism matters more than just the backbone.

Block Type	Gflops	Rel. FID
Cross-Attention	137.6	High
In-Context	119.4	Very High
adaLN	118.6	Medium
adaLN-Zero	**118.6**	**Lowest**

AdaLN-Zero's identity initialization is essential for its superior performance.

Conclusion: All subsequent results use the $\mathbf{adaLN-Zero}$ DiT block.

The DiT Scaling Law: Gflops are Critical

Hypothesis: Increasing model compute (Gflops) should increase sample quality.

Key Scaling Variables Explored:

- **Model Size (Depth/Width):** Increase Gflops, increase parameters.
- **Patch Size (p):** Decrease p, $\rightarrow$ Increase Sequence Length T, $\rightarrow$ **Increase Gflops**, $\rightarrow$ **Parameters are fixed**.

Finding: Gflops vs. Parameters

- Holding model size constant while decreasing patch size improves FID considerably.
- Since parameters are fixed, this proves that additional **model compute (Gflops)** is the key to improved performance, not just parameter count.

Correlation

- The study found a strong negative correlation ($-\mathbf{0.93}$) between Transformer Gflops and FID-50K.
- DiT models with similar **Gflops** achieve similar **FID** values, regardless of whether that Gflops came from depth/width or longer sequences (smaller patch size).

Conclusion: Increasing Gflops (by scaling size or tokens) consistently yields better generative models.

Results

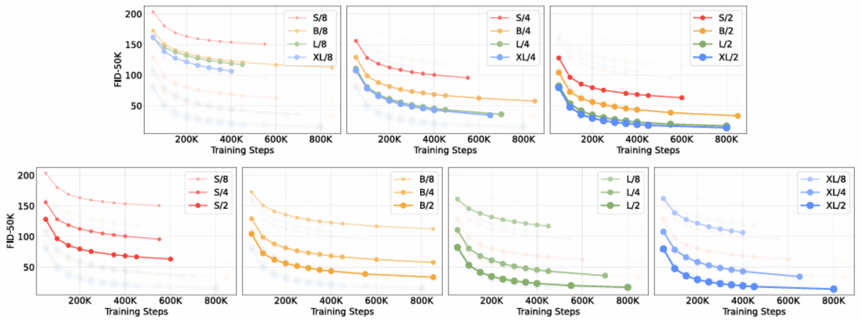


Figure 6. **Scaling the DiT model improves FID at all stages of training.** We show FID-50K over training iterations for 12 of our DiT models. *Top row:* We compare FID holding patch size constant. *Bottom row:* We compare FID holding model size constant. Scaling the transformer backbone yields better generative models across all model sizes and patch sizes.

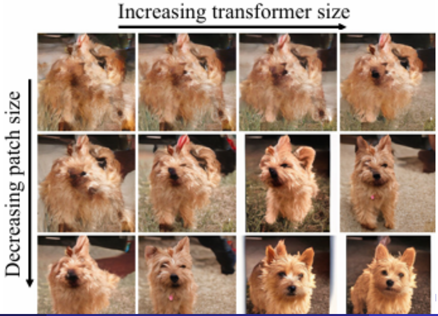
Scaling Experiments: Gflops Drive Quality

Core Finding: Increasing model compute (Gflops) consistently improves image quality (lowers FID).

1. **Scaling Model Size (Fixed Patch p) $\uparrow$**
   - **Experiment:** Hold patch size p constant (e.g., $p = 4$). Increase size from S $\rightarrow$ XL.
   - **Effect:** Transformer becomes deeper and wider (increases parameters & Gflops).
   - **Result (Figure 6, top):** Significant improvements in **FID** are obtained over all stages of training by making the transformer deeper and wider.

2. **Scaling Patch Size (Fixed Model Size)**
   - **Experiment:** Hold model size constant (e.g., DiT-L). Decrease patch size ($p = 8 \rightarrow 2$).
   - **Effect: Sequence Length** (**T**) increases, $\rightarrow$ **Gflops** increase. Parameters remain approximately fixed.
   - **Result (Figure 6, bottom): Considerable FID improvements** are observed throughout training by simply scaling the number of tokens processed.

Conclusion: The results from experiment 2 prove that scaling **model Gflops** is the critical ingredient for improved performance, rather than parameter count alone.


Increasing transformer size
Decreasing patch size

Model	Image Resolution	Flops (G)	Params (M)	Training Steps (K)	Batch Size	Learning Rate	DiT Block	FID-50K (no guidance)
DiT-S/8	256×256	0.36	33	400	256	1×10^{-4}	adaLN-Zero	153.60
DiT-S/4	256×256	1.41	33	400	256	1×10^{-4}	adaLN-Zero	100.41
DiT-S/2	256×256	6.06	33	400	256	1×10^{-4}	adaLN-Zero	68.40
DiT-B/8	256×256	1.42	131	400	256	1×10^{-4}	adaLN-Zero	122.74
DiT-B/4	256×256	5.56	130	400	256	1×10^{-4}	adaLN-Zero	68.38
DiT-B/2	256×256	23.01	130	400	256	1×10^{-4}	adaLN-Zero	43.47
DiT-L/8	256×256	5.01	459	400	256	1×10^{-4}	adaLN-Zero	118.87
DiT-L/4	256×256	19.70	458	400	256	1×10^{-4}	adaLN-Zero	45.64
DiT-L/2	256×256	80.71	458	400	256	1×10^{-4}	adaLN-Zero	23.33
DiT-XL/8	256×256	7.39	676	400	256	1×10^{-4}	adaLN-Zero	106.41
DiT-XL/4	256×256	29.05	675	400	256	1×10^{-4}	adaLN-Zero	43.01
DiT-XL/2	256×256	118.64	675	400	256	1×10^{-4}	adaLN-Zero	19.47
DiT-XL/2	256×256	119.37	449	400	256	1×10^{-4}	in-context	35.24
DiT-XL/2	256×256	137.62	598	400	256	1×10^{-4}	cross-attention	26.14
DiT-XL/2	256×256	118.56	600	400	256	1×10^{-4}	adaLN	25.21
DiT-XL/2	256×256	118.64	675	2352	256	1×10^{-4}	adaLN-Zero	10.67
DiT-XL/2	256×256	118.64	675	7000	256	1×10^{-4}	adaLN-Zero	9.62
DiT-XL/2	512×512	524.60	675	1301	256	1×10^{-4}	adaLN-Zero	13.78
DiT-XL/2	512×512	524.60	675	3000	256	1×10^{-4}	adaLN-Zero	11.93

Table 4. **Details of all DiT models.** We report detailed information about every DiT model in our paper. Note that FID-50K here is computed *without* classifier-free guidance. Parameter and flop counts exclude the VAE model which contains 84M parameters across the encoder and decoder. For both the 256×256 and 512×512 DiT-XL/2 models, we never observed FID saturate and continued training them as long as possible. Numbers reported in this table use the ft-MSE VAE decoder.

Final Results: State-of-the-Art Performance

DiT-XL/2 achieves the lowest FID on ImageNet, proving Transformer scaling success.

1. **ImageNet** 256×256 **Comparison**
   - **Model:** DiT-XL/2-G (cfg=1.5).
   - **Compute Efficiency:** DiT-XL/2 (118.6 Gflops) is far more efficient than pixel-space ADM (1120 Gflops).

2. **ImageNet** 512×512 **Comparison**
   - **Model:** DiT-XL/2-G (cfg=1.5).
   - **Compute:** DiT-XL/2 uses 524.6 Gflops vs. ADM-U's 2813 Gflops.

Model Previous SOTA	FID-50K
StyleGAN-XL	2.30
LDM-4-G	3.60
DiT-XL/2-G	**2.27**

Model Previous SOTA	FID-50K ↓
StyleGAN-XL	2.41
ADM-G,ADM-U	3.85
DiT-XL/2-G	**3.04**

Conclusion: Scaling the Transformer backbone yields the best generative models across all major benchmarks and resolutions.

Conclusion: The Legacy of Diffusion Transformers

DiT established a new foundation for generative modeling by proving the scalability of Transformers in the diffusion process.

1. **Architectural Paradigm Shift**
   - The paper introduced the concept of replacing the inductive bias-heavy **U-Net** with a scalable **Transformer** backbone.
   - This demonstrated that we can successfully transfer the scaling laws of Transformers (ViT/LLMs) to the image generation domain.

2. **The Fundamental Scaling Law**
   - Increasing model compute (**Gflops**) through **depth, width,** or **sequence length (smaller patch size)** directly improves output quality.
   - **Gflops** are the critical ingredient for performance, proven by a strong $-\mathbf{0.93}$ correlation with **FID** (Sample Quality).

3. **The Technical Key: AdaLN-Zero**
   - The success of scaling was contingent on the efficient **AdaLN-Zero** block design.
   - This mechanism provides superior conditioning performance and stability due to its specialized prediction MLP and **zero-initialized residual gate ($\alpha = 0$)**.

4. **State-of-the-Art (SOTA) Result**
   - The scaled DiT-XL/2 achieved the **best FID** on the ImageNet benchmark, surpassing all prior generative models (LDM, ADM, StyleGAN-XL).

Sample

```python
from IPython.display import Image
# Replace the path below with the actual location of your generated file
Image(filename='/content/sample.png')
```

Backup on diffusion

Making the Posterior Tractable (Bayes' Rule)

Attempt 1: Standard Bayes Rule

$$q(x_{t-1}|x_t) = \frac{q(x_t|x_{t-1})q(x_{t-1})}{q(x_t)}$$

Problem: The marginals $q(x_t)$ are intractable (requires integrating all possible images).

Attempt 2: Conditioning on x_0 (The Trick) By conditioning on the start image x_0, we use Bayes rule relative to x_0:

$$q(x_{t-1}|x_t, x_0) = \frac{q(x_t|x_{t-1}, x_0) \cdot q(x_{t-1}|x_0)}{q(x_t|x_0)}$$

Using the Markov property (x_t depends only on x_{t-1}):

$$q(x_{t-1}|x_t, x_0) = \frac{\underbrace{q(x_t|x_{t-1})}_{\text{Forward Step}} \cdot \underbrace{q(x_{t-1}|x_0)}_{\text{Jump from } x_0}}{\underbrace{q(x_t|x_0)}_{\text{Jump from } x_0}}$$

The Tractable "Ground Truth"

Since we are multiplying/dividing Gaussians, the result is a Gaussian:

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t \mathbf{I})$$

After algebraic manipulation, the mean $\tilde{\mu}_t$ is:

$$\tilde{\mu}_t(x_t, x_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} x_t$$

The Training Signal

Our neural network $p_\theta(x_{t-1}|x_t)$ tries to predict **this specific mean**. We substitute $x_0 \approx (x_t - \sqrt{1-\bar{\alpha}_t}\epsilon)/\sqrt{\bar{\alpha}_t}$ to train on predicting the noise ϵ instead.

The "Trick": Reparameterization

Forward Process: We add noise incrementally: $x_t = \sqrt{1-\beta_t} x_{t-1} + \sqrt{\beta_t}\epsilon$.

The Closed-Form Trick: Instead of iterating t times, we can sample x_t directly from x_0. Let $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod \alpha_i$:

Reparameterization Property

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$$

- **Why?** This allows efficient random access to any timestep t during training without running a loop.

The Mathematics of the Update

Where did that update equation come from? It comes from the true posterior mean conditioned on x_0:

$$\tilde{\mu}_t(x_t, x_0) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon \right)$$

The Substitution: The model p_θ doesn't know ϵ, so we replace it with the network prediction $\epsilon_\theta(x_t, t)$.

$$\mu_\theta(x_t, t) = \underbrace{\frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right)}_{\text{Denoising Term}}$$

We add $\sigma_t z$ (Langevin noise) to match the variance of the distribution.

$$x_{t-1} = \underbrace{\mu_\theta(x_t)}_{\text{Predicted Image}} + \underbrace{\sigma_t \cdot z}_{\text{Random Noise}}$$

The Statistical Goal & Intractability

Objective: Maximize the likelihood of the training data x_0.

$$\max_{\theta} \mathbb{E}_{x_0 \sim q_{data}}[\log p_\theta(x_0)]$$

The Problem: Intractability

To calculate the true likelihood, we must marginalize over all latent steps:

$$p_\theta(x_0) = \int p_\theta(x_0, x_1, \ldots, x_T)\, dx_{1:T}$$

This integral is **intractable** because we cannot evaluate all possible paths the noise could take.

The Missing Piece: We need the true reverse posterior $q(x_{t-1}|x_t)$ to guide training, but it is also intractable without knowing x_0.

Derivation: From Likelihood to KL Divergence

Step 1: Introduction of q (The "Multiply by 1" Trick) We introduce the intractable latent states $x_{1:T}$ and the variational posterior $q(x_{1:T}|x_0)$.

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

Step 2: Jensen's Inequality (The Lower Bound) Moving the log inside the expectation ($\log \mathbb{E} \geq \mathbb{E} \log$):

$$\log p_\theta(x_0) \geq \underbrace{\mathbb{E}_q \left[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)} \right]}_{\text{Evidence Lower Bound (ELBO)}}$$

Step 3: Rearranging to KL Divergence Splitting the log term reveals the KL Divergence structure:

$$\begin{aligned} \text{ELBO} &= \mathbb{E}_q[\log p_\theta(x_{0:T})] - \mathbb{E}_q[\log q(x_{1:T}|x_0)] \\ &\approx -\sum_{t=1}^{T} D_{KL}(q(x_{t-1}|x_t, x_0) \,||\, p_\theta(x_{t-1}|x_t)) \end{aligned}$$

The Mathematical Bridge (VLB)

Since $q(x_{t-1}|x_t)$ is intractable, we switch the teacher from $q(x_{t-1}|x_t)$ to the **conditioned posterior** $q(x_{t-1}|x_t, x_0)$, which is tractable.

The objective changes from maximizing Likelihood to minimizing the Upper Bound on the negative log-likelihood:

$$\mathcal{L}_{\text{VLB}} = \mathbb{E}_q \left[\sum_{t=1}^{T} D_{KL}\big(\underbrace{q(x_{t-1}|x_t, x_0)}_{\text{Tractable Teacher}} \;||\; \underbrace{p_\theta(x_{t-1}|x_t)}_{\text{Student Model}} \big) \right]$$

- **Teacher:** Can solve the reverse step because it sees the answer key (x_0).
- **Student:** Must approximate the Teacher without seeing x_0.

From Distributions to Geometry

Both the Teacher (q) and the Student (p_θ) are **Gaussian Distributions**.

- Teacher: $q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}(x_t, x_0), \tilde{\beta} I)$
- Student: $p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t), \Sigma_\theta(x_t))$

Minimizing KL Divergence between two Gaussians simplifies to minimizing the **Squared Euclidean Distance** between their means:

The Geometric Loss

$$D_{KL} \approx ||\tilde{\mu}(x_t, x_0) - \mu_\theta(x_t)||^2 + C$$

We simply want the Model's Mean (μ_θ) to match the True Posterior Mean ($\tilde{\mu}$).

The Final Form (Reparameterization)

The mean μ is strictly defined by the current state x_t and the noise ϵ.

$$\text{Teacher Mean: } \tilde{\mu}(x_t, x_0) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_t\right)$$

$$\text{Student Mean: } \mu_\theta(x_t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t)\right)$$

Substituting these into the loss equation, the x_t terms cancel out, leaving only the noise terms:

Final Objective: L_{simple}

$$||\epsilon_t - \epsilon_\theta(x_t)||^2$$

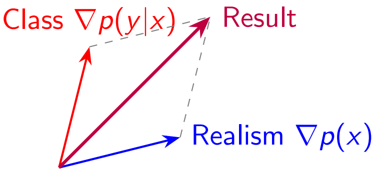
"Predicting the mean is mathematically identical to predicting the noise."

Classifier Guidance: The "Artist" and the "Guide"

Before Classifier-Free Guidance, we used two separate models.

1. The Artist (Diffusion Model): Computes $\nabla_x \log p(x)$. Knows how to make images look *real*, but doesn't know classes.

2. The Guide (Classifier): Computes $\nabla_x \log p(y|x)$. Knows how to make images look like *class* y.



The Update Rule

We shift the predicted mean μ_θ using the classifier gradient:

$$\hat{\mu}(x_t) = \mu_\theta(x_t) + s \cdot \Sigma_\theta(x_t) \nabla_{x_t} \log p_\phi(y|x_t)$$

Why we need a "Noisy" Classifier

We must train a **specialized classifier** $p_\phi(y|x_t)$ that:

1. Takes noisy input x_t.
2. Takes the timestep t as input.
3. Is robust enough to guide generation from pure noise.

This complexity (maintaining two pipelines) led to the adoption of Classifier-Free Guidance.

Classifier-Free Guidance: The Dual-Prediction Trick

Goal: Train a single generative model (ϵ_θ) to produce both conditional ($\mathbf{c}$) and unconditional ($\emptyset$) outputs.

1. Training: The Dropout Mechanism

- The model (U-Net/DiT) uses Adaptive Norm (AdaGN/AdaLN) to receive condition $\mathbf{c}$.
- Random Dropout (10-20%): During training, the condition vector $\mathbf{c}$ is randomly replaced by a **Null Embedding** ($\emptyset$).

Resulting Behaviors Learned:

1. **When $\mathbf{c} \neq \emptyset$:** Learns the Conditional Path (ϵ_{cond}), using the prompt to steer the features.
2. **When $\mathbf{c} = \emptyset$:** Learns the Unconditional Path (ϵ_{uncond}), predicting the noise for a generic realistic image.

The single set of weights encodes both "knowledge of specific content" and "knowledge of realism."

2. Inference: Executing the Dual Pass At sampling time, the model is run twice with the **same noisy input $\mathbf{x}_t$** but with different conditions:

Condition	Output	CFG Role
$\mathbf{c}$ (Prompt)	ϵ_{cond}	Direction (Spec
$\emptyset$ (Null)	ϵ_{uncond}	Realism

The Final Steer: The two noise vectors are combined mathematically to generate the final, steered noise prediction ($\hat{\epsilon}$), where $s > 1$ is the guidance scale:

$$\hat{\epsilon} = \epsilon_{\text{uncond}} + s \cdot (\epsilon_{\text{cond}} - \epsilon_{\text{uncond}})$$

Loss Hybridization: Training the Variance ($\boldsymbol{\Sigma}_\theta$)

Principle: The simplification of $\mathbf{D}_{KL}$ to MSE only holds if variance is fixed.

Case 1: Fixed Variance (Original DDPM)

- True $\boldsymbol{\Sigma}_P$ = Model $\boldsymbol{\Sigma}_Q$ = constant.
- Variance terms in the KL divergence cancel out or become constant.
- The remaining loss is proportional to the difference in means:

$$\mathcal{L}_{\text{VLB}} \propto (\boldsymbol{\mu}_P - \boldsymbol{\mu}_Q)^2 \equiv \mathcal{L}_{\text{simple}}$$

Case 2: Learned Variance (DiT / IDDPM)

- Model $\boldsymbol{\Sigma}_\theta$ is predicted by the network ($\boldsymbol{\Sigma}_Q = \boldsymbol{\Sigma}_\theta$).
- The full KL divergence contains terms depending on $\boldsymbol{\Sigma}_\theta$:

$$\mathbf{D}_{KL} \propto \underbrace{\log\left(\frac{\boldsymbol{\Sigma}_\theta}{\boldsymbol{\Sigma}_P}\right)}_{\text{Log-Ratio Term}} + \frac{(\boldsymbol{\mu}_P - \boldsymbol{\mu}_Q)^2}{2\boldsymbol{\Sigma}_\theta}$$

Conclusion: The $\mathcal{L}_{\text{VLB}}$ is required because only the Log-Ratio and Inverse Variance terms provide the gradient signal to train $\boldsymbol{\Sigma}_\theta$ (noise part is trained with one loss and the covar part with a different loss, hence the different info).

Hybrid Loss Strategy (DiT/IDDPM): The total loss $\mathcal{L}_{\text{DiT}} = \mathcal{L}_{\text{simple}} + \lambda \cdot \mathcal{L}_{\text{VLB}}$ is used.

- $\mathcal{L}_{\text{simple}}$ trains the **Mean** ($\boldsymbol{\mu}_\theta$ / noise ϵ_θ) for high visual quality.
- $\mathcal{L}_{\text{VLB}}$ trains the **Variance** ($\boldsymbol{\Sigma}_\theta$) for mathematical fidelity and faster sampling.

$$L_{\text{VLB}} = \mathbb{E}_q[\underbrace{D_{KL}(q(x_T|x_0)||p(x_T))}_{L_T\text{:Constant Prior Match}} + \sum_{t=2}^{T} \underbrace{D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))}_{L_{t-1}\text{:\textbf{The Denoising Term}}} \underbrace{-\log p_\theta(x_0|x_1)}_{L_0\text{:Reconstruction}}]$$

Backup DiT

Model	Layers N	Hidden size d	Heads	Gflops $(I=32,\ p=4)$
DiT-S	12	384	6	1.4
DiT-B	12	768	12	5.6
DiT-L	24	1024	16	19.7
DiT-XL	28	1152	16	29.1

Efficiency: Latent Diffusion Models (LDM)

The Problem: Training directly on high-res pixels (512×512) is computationally prohibitive.
The Solution (Two-Stage Approach): Move the diffusion process into a compressed **Latent Space**.

- **Stage 1 (Perceptual):** Train an Autoencoder (E, D) to learn latents $z = E(x)$; then freeze it.
- **Stage 2 (Semantic):** Train the diffusion model entirely in latent space (small spatial dimensions).
- **DiT Strategy:** Stage 1 uses a convolutional VAE; Stage 2 replaces the U-Net with a **Transformer (DiT)**.

Conditional Probability vs. Parameterization

The Dilemma: The model must represent a probability, but we only have functions.

1. **The Ideal: Conditional Probability** (Goal)
   - We must model the reverse step conditioned on class $\mathbf{c}$:

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{c})$$

   - Since noise steps are Gaussian, the distribution is defined entirely by:
     - $\boldsymbol{\mu}_\theta(\mathbf{x}_t, \mathbf{c})$ (Mean)
     - $\boldsymbol{\Sigma}_\theta(\mathbf{x}_t, \mathbf{c})$ (Variance)

The entire challenge is making $\boldsymbol{\mu}_\theta$ and $\boldsymbol{\Sigma}_\theta$ conditional on $\mathbf{c}$.

2. **The Parameterization: AdaLN** (Solution)
   - The DiT Transformer is the function that predicts the mean $\boldsymbol{\mu}_\theta$.
   - The **AdaLN mechanism** is the architectural solution that forces the function to be conditional:

AdaLN as the Link

The AdaLN MLP ensures the function $\boldsymbol{\mu}_\theta$ correctly incorporates $\mathbf{c}$ by using it to dynamically gate features:

$$\boldsymbol{\mu}_\theta \text{ depends on } \mathbf{x}_t \text{ and } \mathbf{c}$$

AdaLN guarantees a robust and efficient path for $\mathbf{c}$ to control the predicted Gaussian parameters.

Efficiency: Larger Models Are More Compute-Efficient

Question: Does training a small model for longer steps equal a large model for fewer steps?

Finding 1: Scaling is Visually Effective

- Visual inspection of samples (Figure 7) confirms that increasing Gflops (size or tokens) leads to notable improvements in visual fidelity and detail.
- Scaling the number of tokens (smaller p) yields significant quality improvements.

Finding 2: Compute Efficiency (Figure 9)

- Larger DiT models use total training compute more efficiently.
- Small models, even when trained for a long time, eventually become compute-inefficient relative to larger models.
- Example: DiT-XL/4 significantly outperforms DiT-XL/2 after 10^{10} total training Gflops.

Implication: When planning large-scale training runs, investing compute into a **larger model** is a better use of resources than training a smaller model for more steps.

Implementation Deep Dive & CFG Ablation

Focus: Specifics of conditioning (AdaLN) and Guidance (CFG subsetting).

1. Conditioning Mechanism (AdaLN/Timestep)

- **Timestep t:** Embedded using a **256**-dim frequency embedding, followed by a **2**-layer MLP with **SiLU** activations.
- **Core Transformer:** Uses **GELU** activations in the FFNs.
- **AdaLN Output Size:** The linear layer predicting $\gamma, \beta, (\alpha)$ outputs different sizes:
  - **AdaLN:** $4\times$ hidden size.
  - **AdaLN-Zero:** **6**$\times$ hidden size (to include the α gate).

2. CFG Ablation: Subsetting Guidance

- **Method:** CFG was applied only to the **first three channels** of the 4-channel latent space, instead of all four.
- **Result:** Subset guidance yields **similar FID results** to full guidance when the scale is adjusted.
- **Scale Approximation:** The relationship is roughly linear:

$$s_{3\text{ch}} = (1 + x) \approx s_{4\text{ch}} = (1 + \frac{3}{4}x)$$

- **Implication:** Good performance can be achieved by applying guidance to a subset of latent elements, suggesting robustness in the latent space structure.

Scaling Results: Generalization Beyond FID

Question: Do Gflops improve metrics other than FID? (Yes.)

1. **Impact on Evaluation Metrics**
   - The high correlation between **Model Gflops** and performance holds across **every** metric.
   - **Metrics:** sFID, Inception Score (IS), Precision, and Recall all improve with scale.
   - Specific Benefit: Inception Score and Precision benefit particularly heavily from increased model scale.

2. **Impact on Training Stability**
   - Scaling DiT Gflops (via size or tokens) causes the training loss to **decrease more rapidly**.
   - The loss also **saturates** at a lower value (Figure 13).
   - **LLM Consistency:** This loss curve behavior is consistent with trends observed in scaled-up Transformer-based language models.

VAE Decoder Ablations: Robustness Check

Question: Does DiT's performance rely on a specific VAE decoder? (No.)

1. **Experimental Setup**
   - **Goal:** Test if DiT's superior FID is dependent on VAE fine-tuning.
   - **Test Decoders:** Tested the **original** LDM decoder and two Stable Diffusion fine-tuned decoders ($\mathbf{ft-MSE}, \mathbf{ft-EMA}$).
   - **Decoupling:** Because the encoders were identical, the decoders were swapped in **without retraining** the DiT diffusion model.

2. **Results (DiT-XL/2, cfg=1.5)**

Decoder	**FID ↓**	**IS ↑**
Original LDM	2.46	271.56
ft-MSE	2.30	276.09
ft-EMA	**2.27**	**278.24**

Table: Performance with different VAE Decoders.

Conclusion:

- The different pre-trained decoders yielded comparable results.
- The DiT model's SOTA performance is robust and due to the **Transformer backbone** (DiT), not specialized decoder tuning.

Understanding the "Mean" in Diffusion (1/2)

Misconception: The "Mean" is the average color of the whole image (e.g., Gray).

Reality: The Diffusion Model is a **Conditional Probability Machine**.

- It does not ask: *"What is the average pixel?"*
- It asks: *"What is the average value of the top-left pixel,* **GIVEN** *that the bottom-right pixel is a cat's tail?" (done with the attention framework)*

Context Determines the Mean

Because the answer depends on the context (x_t), the mean changes across the image:

- **Pixel at Eye:** Highest probability value → **Black**.
- **Pixel at Tooth:** Highest probability value → **White**.
- **Pixel at Sky:** Highest probability value → **Blue**.

The "Mean" is simply the mathematical term for "The Model's Prediction of the Image", gaussian mean is the highest point of probability.

The Mathematical View: Multivariate Gaussian (2/2)

How do individual pixel means form a "Process Mean"?

The Image as a Point

An image is treated as a single point in a high-dimensional space (e.g., 1,024 dimensions for 32×32).

$$\text{Denoising Process} \sim \mathcal{N}(x_{t-1}; \mu_\theta, \Sigma_\theta)$$

Connecting Coordinates to Pixels:

- The **Mean of the Process (process multivariate gaussian** is a Vector: $\mu_\theta = [\mu_1, \mu_2, \ldots, \mu_{1024}]$.
- The **Mean of a Pixel** is just one coordinate in that list.

Summary

Satisfying the "Mean of the Process" (Global)

$\equiv$

Predicting the correct "Mean for every Pixel" (Local Coordinates).