

Lecture 16:

Generative Models - Diffusion

Ho-min Park

homin.park@ghent.ac.kr

powersimmani@gmail.com

Lecture Contents

Part 1: Introduction and Motivation

Part 2: Forward Process

Part 3: Reverse Process

Part 4: Sampling

Part 5: Architecture

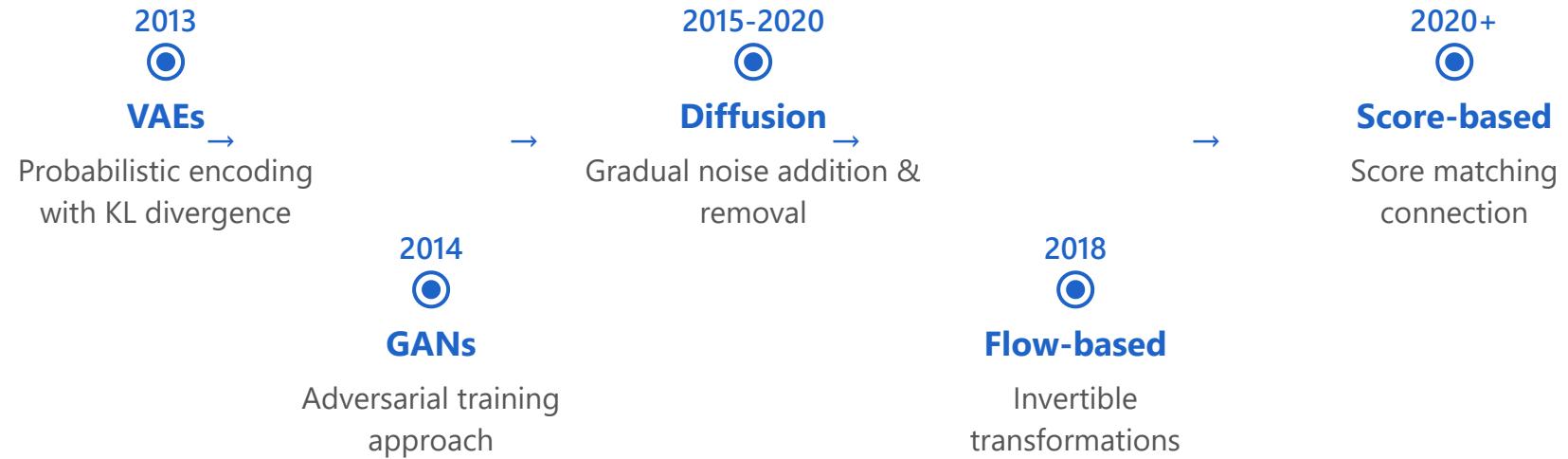
Part 6: Advanced Techniques

Part 7: Applications and Extensions

Part 1/7:

Introduction & Motivation

- 1.** Evolution of Generative Models
- 2.** Intuitive Understanding
- 3.** Comparison with GANs



Recent Success (2022-2023)

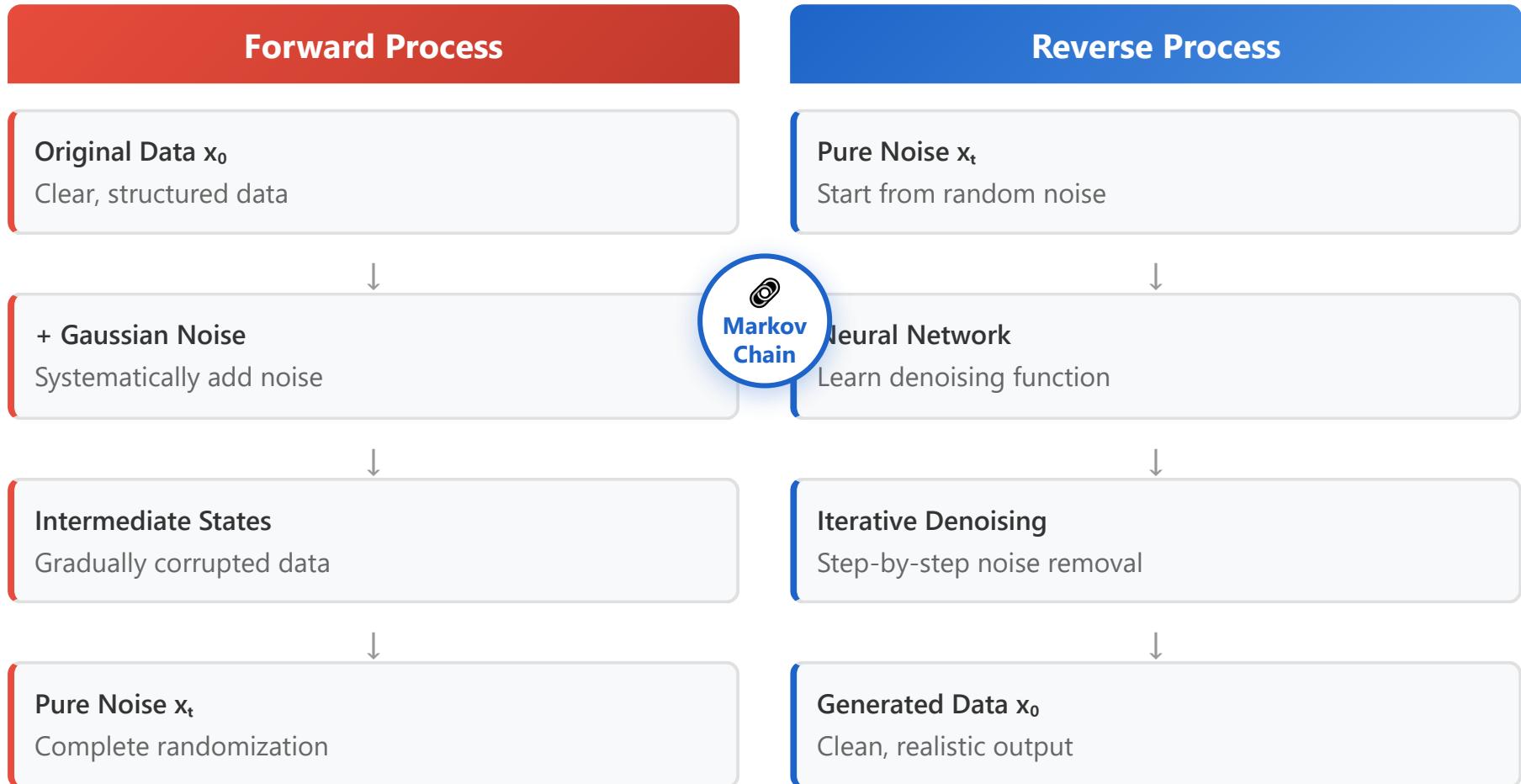
DALL-E 2

Stable Diffusion

Midjourney

Key Advantages: High quality • Stable training • Mode coverage

 Like a sculptor gradually refining marble from rough block to statue



Key Insights

- ✓ Easier to learn small denoising steps
- ✓ Each step depends only on previous
- ✓ More controllable than direct generation



Diffusion Models vs GANs

Comprehensive Feature Comparison

Criterion	Diffusion Models	GANs
Training Stability	More stable, no adversarial collapse ✓	Can be unstable, risk of mode collapse
Mode Coverage	Captures all modes effectively ✓	May miss some modes
Sample Quality	Comparable, often more detailed ✓	High quality, sometimes less detailed
Generation Speed	Slower (100-1000 steps)	Faster (1 step generation)
Training Complexity	Simpler loss, no discriminator ✓	Complex, requires discriminator tuning

Advantage ✓ Winner in this category

Part 2/7:

Forward Process

- 4.** Forward Process Definition
- 5.** Noise Schedule
- 6.** Forward Process Properties
- 7.** Reparameterization Trick
- 8.** Mathematical Foundation

Forward Process Definition

Part 2/7: Forward Process



Goal: Gradually transform data into pure Gaussian noise



Markov Chain: $q(x_t | x_{t-1})$

x_0

Original Data

$t = 1, 2, \dots, T$



x_t

$\sim N(0, I)$



Gaussian Transition (Distribution Form)

$$q(x_t | x_{t-1}) = N(x_t; \sqrt{\alpha_t} \cdot x_{t-1}, (1-\alpha_t)I) \text{ where } \alpha_t = 1 - \beta_t$$



Sampling Form (Reparameterization)

$$x_t = \sqrt{\alpha_t} \cdot x_{t-1} + \sqrt{(1-\alpha_t)} \cdot \varepsilon, \text{ where } \varepsilon \sim N(0, I)$$

Noise Schedule (β_t)

Controls amount of noise at step t

Fixed Schedule

β_t increases from 0.0001 to 0.02

Time Steps (T)

Usually $T = 1000$ steps

End Result

$x_t \sim N(0, I)$ is pure Gaussian noise

Cumulative Effect

Part 2/7: Forward Process

✨ Closed Form Solution

Jump directly from x_0 to any x_t without iterating through all steps

✗ Iterative (Slow)



Need all intermediate steps

✓ Direct Sampling (Fast)



One-step computation

Mathematical Formulation

Alpha definitions:

$$\alpha_t = 1 - \beta_t$$

Cumulative product:

$$\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$$

Direct Sampling Formula:

$$x_t = \sqrt{\bar{\alpha}_t} \cdot x_0 + \sqrt{1 - \bar{\alpha}_t} \cdot \epsilon$$



Efficient Training



Signal-to-Noise Control



Reparameterization

Mathematical Properties

Part 2/7: Forward Process



Mean Preservation

$$E[x_t] = \sqrt{\alpha_t} \cdot x_0$$

Expected value scaled by $\sqrt{\alpha_t}$



Variance Growth

$$\text{Var}[x_t] = (1-\alpha_t) \cdot I$$

Variance increases as noise accumulates



Gaussian Distribution

$$q(x_t | x_0) = N(x_t; \sqrt{\alpha_t} \cdot x_0, (1 - \alpha_t) \cdot I)$$

Complete distributional characterization



Posterior Distribution

$$q(x_{t-1} | x_t, x_0)$$

Also Gaussian with closed-form solution



Tractable Likelihood

Can compute exact log-likelihood for model evaluation and comparison



Reversibility

Theoretical foundation enables the reverse generative process



Noise Independence

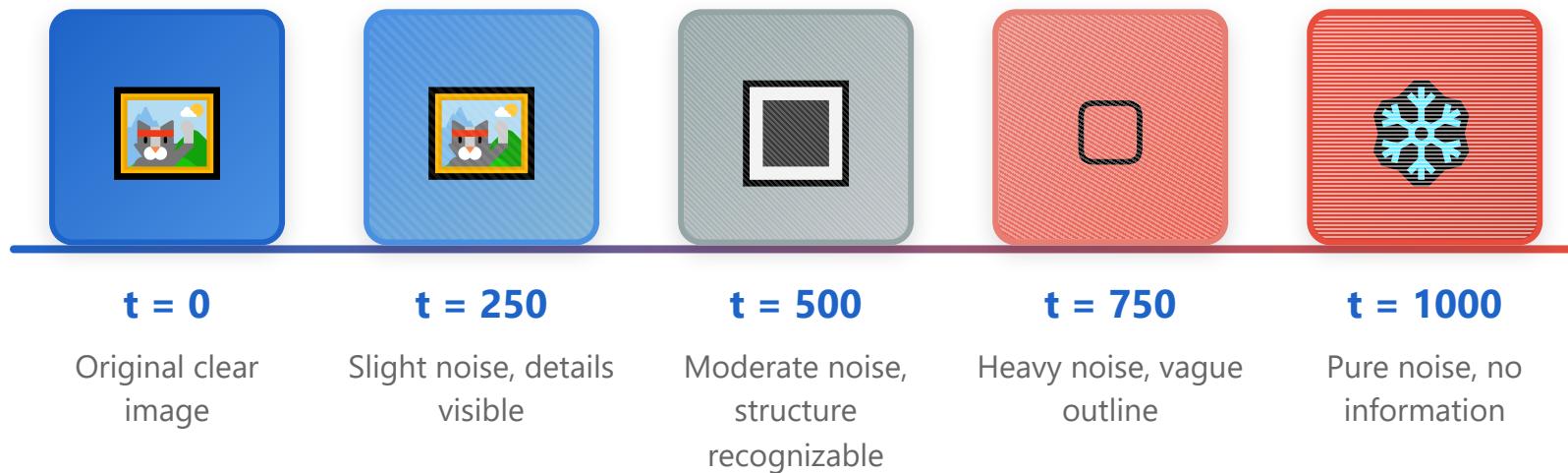
Noise at different timesteps is independent,
simplifying analysis



All properties maintain Gaussian structure, enabling tractable inference and generation

Forward Process Visualization

Part 2/7: Forward Process



Information Loss

Gradual and controlled degradation over timesteps

Visual Inspection

Validates noise schedule effectiveness

Why is this Process Necessary?

Part 2/7: Forward Process



Understanding the Fundamental Motivation



Tractable Posterior

Known distribution $q(x_{t-1}|x_t, x_0)$ enables efficient training



Curriculum Learning

Start easy (small noise), gradually increase difficulty



Stable Gradients

Small steps ensure smooth gradient flow during training



Mode Coverage

Noise prevents collapse to single mode



Theoretical Foundation

Enables variational bound derivation for optimization



Flexible Design

Can adjust β_t schedule for different data types



Reversibility

Ensures reverse generative process is well-defined



The forward process creates a tractable learning framework that makes the reverse process trainable

Part 3/7:

Reverse Process

- 9.** Reverse Process Goal
- 10.** Neural Network Parameterization
- 11.** Deep Dive into Score Function
- 12.** Denoising Objective Function
- 13.** Training Algorithm

Reverse Process Goal

Part 3/7: Reverse Process



Objective: Learn to reverse the forward diffusion process

Start Point

$$\mathbf{x}_T \sim N(0, I)$$

Pure Noise

$$p_{\theta}(\mathbf{x}_{t-1} | \mathbf{x}_t)$$

Iterative Denoising



End Point

$$\mathbf{x}_0 \sim p_{\text{data}}(\mathbf{x})$$

Data Sample



Markov Chain

Parameterized by neural network



Conditional Distribution

Model Gaussian transitions



Denoising Direction

Remove added noise

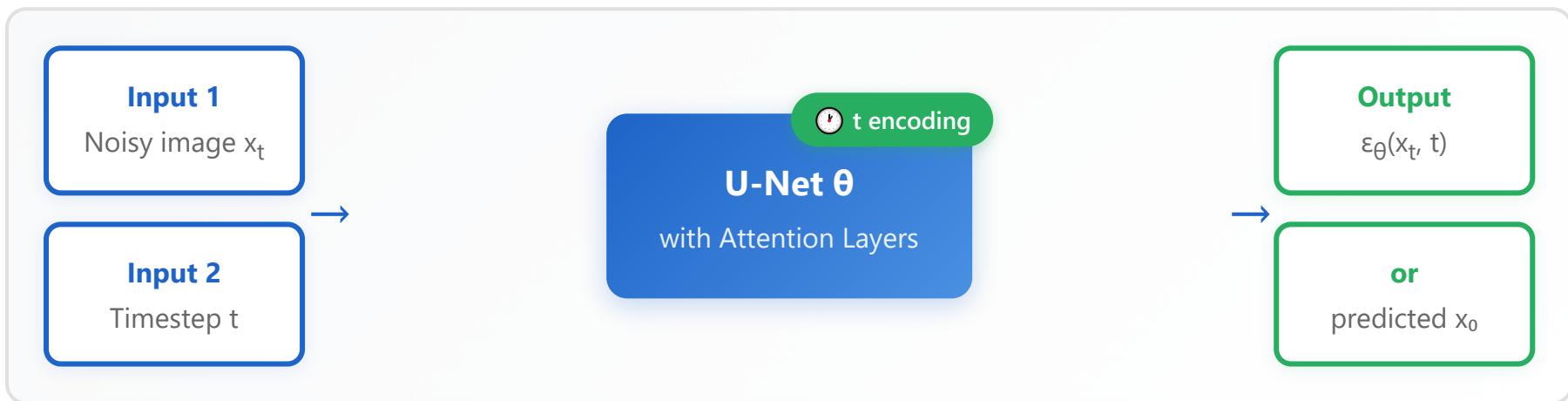


Generation Process

Iteratively denoise from noise to create new high-quality samples

Neural Network Parameterization

Part 3/7: Reverse Process



Mean Prediction

$$\mu_\theta(x_t, t) = 1/\sqrt{\alpha_t} \cdot (x_t - (1-\alpha_t)/\sqrt{1-\bar{\alpha}_t}) \cdot \varepsilon_\theta(x_t, t)$$

Variance

$$\sigma_t^2 \text{ (Fixed or Learned)}$$

Reverse Transition Distribution

$$p_\theta(x_{t-1} | x_t) = N(x_{t-1}; \mu_\theta(x_t, t), \sigma_t^2 I)$$

Deep Dive into Score Function

Part 3/7: Reverse Process

Score Definition

$$\nabla_{\mathbf{x}} \log p(\mathbf{x})$$

Indicates direction of density increase

Connection

Denoising \approx Score matching

Tweedie's Formula

Score relates to optimal denoising estimation

Score Estimation

$$\varepsilon_{\theta}(\mathbf{x}_t, t) \approx -\sqrt{1-\bar{\alpha}_t} \cdot \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)$$

Langevin Dynamics

Reverse process as Langevin sampling from the score function

Continuous Limit

Score-based SDEs when $T \rightarrow \infty$



Unified View

Diffusion and score-based models are equivalent

Denoising Objective Function

Part 3/7: Reverse Process

Simplified Loss

✓ Preferred

$$L_{\text{simple}} = E_{\{t, x_0, \varepsilon\}} [\|\varepsilon - \varepsilon_\theta(x_t, t)\|^2]$$

Works better in practice

Variational Lower Bound

$$L_{\text{vlb}} = E[\log p(x_0) - \log p_\theta(x_0)]$$

Theoretical foundation

🎯 Practical Training Process

1 Sample Timestep

$t \sim \text{Uniform}\{1, \dots, T\}$

2 Sample Noise

$\varepsilon \sim N(0, I)$

3 Create Noisy Sample

$$x_t = \sqrt{\alpha_t} \cdot x_0 + \sqrt{1-\alpha_t} \cdot \varepsilon$$

4 Gradient Descent

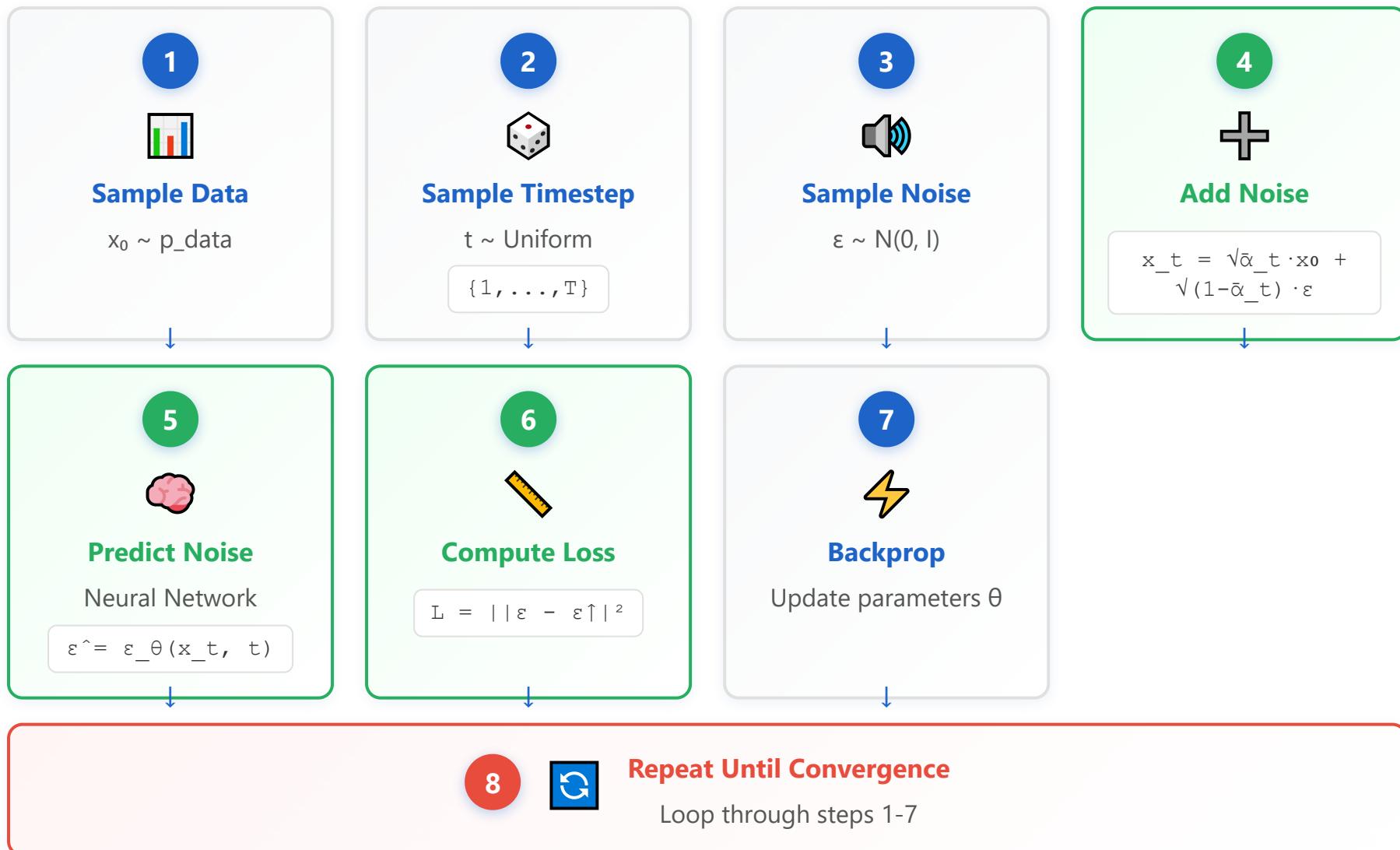
Update θ to minimize loss

⚡ Optimization Objective

$$\text{minimize } \|\varepsilon - \varepsilon_\theta(x_t, t)\|^2$$

Training Algorithm

Part 3/7: Reverse Process



Core Training Loop: Sample → Add Noise → Predict → Optimize → Repeat

Concrete Calculation Example

2D Case with Actual Numbers

📝 Step-by-Step Calculation

Step 1: Sample Data Point

$x_0 = [3.0, 2.0]$

Original data point from training set

Step 2: Sample Timestep

$t = 500$ (out of $T=1000$)

Random timestep, halfway through diffusion

Step 3: Sample Noise

$\varepsilon \sim N(0, I) = [0.8, -1.2]$

Random Gaussian noise vector

Step 4: Add Noise (Forward Process)

$\bar{\alpha}_{500} = 0.5$ (example)

$$x_t = \sqrt{\bar{\alpha}_t} \cdot x_0 + \sqrt{1-\bar{\alpha}_t} \cdot \varepsilon$$

$$x_{500} = \sqrt{0.5} \cdot [3.0, 2.0] + \sqrt{0.5} \cdot [0.8, -1.2]$$

$$x_{500} = [2.12, 1.41] + [0.57, -0.85]$$

$$\text{x500} = [2.69, 0.56]$$

Step 5: Predict Noise

$$\hat{\varepsilon} = \varepsilon_\theta(x_{500}, 500) = [0.75, -1.15]$$

Neural network prediction

Step 6: Compute Loss

$$L = ||\varepsilon - \varepsilon \uparrow||^2$$

$$L = ||[0.8, -1.2] - [0.75, -1.15]||^2$$

$$L = ||[0.05, -0.05]||^2$$

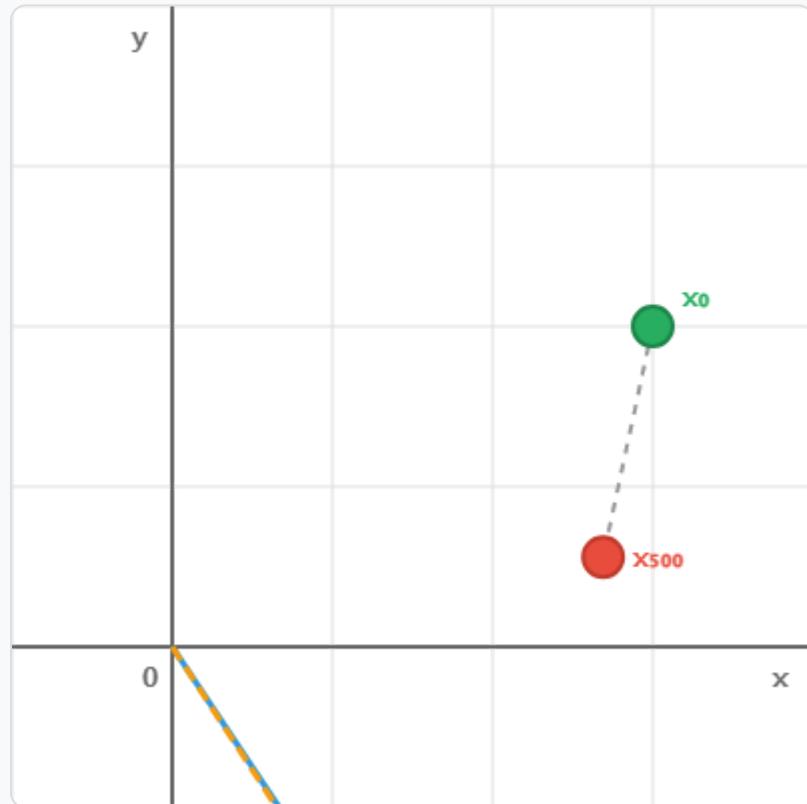
$$L = 0.0025 + 0.0025 = 0.005$$

Step 7: Update Parameters

$$\theta \leftarrow \theta - \alpha \cdot \nabla_{\theta} L$$

Gradient descent to minimize loss

Visual Representation



Legend

- $x_0 = [3.0, 2.0]$ - Original data
- $x_{500} = [2.69, 0.56]$ - Noisy data
- $\varepsilon = [0.8, -1.2]$ - True noise
- $\hat{\varepsilon} = [0.75, -1.15]$ - Predicted noise

Goal: Make $\hat{\varepsilon}$ closer to ε by updating θ

Part 4/7:

Sampling

- 14.** DDPM Sampling
- 15.** DDIM - Fast Sampling
- 16.** Conditional Generation
- 17.** Classifier-Free Guidance

DDPM Sampling

Part 4/7: Sampling

1 Initialize from Pure Noise

$$x_T \sim N(0, I)$$



Iterative Denoising

$t = T, T-1, \dots, 1$

① Predict Noise

$$\hat{\varepsilon} = \varepsilon_\theta(x_t, t)$$

② Compute Mean

$$\mu_\theta(x_t, t)$$

③ Sample Noise

$$z \sim N(0, I) \text{ if } t > 1, \text{ else } z=0$$

④ Update State

$$x_{t-1} = \mu_\theta(x_t, t) + \sigma_t \cdot z$$

3 Final Output

x_0 is the generated sample

 Time Complexity: $O(T)$ forward passes • Typically $T = 1000$

DDIM - Fast Sampling

Part 4/7: Sampling



Problem

DDPM requires many steps (1000) for good quality



Solution

DDIM uses deterministic, non-Markovian process

DDPM

- All timesteps (1000)
- Stochastic process
- Random noise added
- Slower generation

DDIM

⚡ 20x Speedup

- ✓ Subset only (e.g., 50 steps)
- ✓ Deterministic ($\sigma_t = 0$)
- ✓ Skip: $\tau = [1, 20, 40, \dots, 1000]$
- ✓ Comparable quality



DDIM Update Formula

$$x_{t-1} = \sqrt{\alpha_{t-1}} \cdot x_0 + \sqrt{1-\alpha_{t-1}} \cdot \varepsilon_t$$



⚖️ Trade-off: Can interpolate between DDPM and DDIM for speed-quality balance

Conditional Generation

Part 4/7: Sampling



Goal: Generate samples conditioned on class, text, or image

Classifier Guidance Method

Base Score
 s (original)

+

Classifier Gradient
 $w \cdot \nabla_x \log p(y|x_t)$

Modified Score
 \tilde{s}

Modified Score Formula

$$\tilde{s} = s + w \cdot \nabla_x \log p(y|x_t)$$

Guidance Scale

w controls conditioning strength

Applications

- Class-conditional ImageNet generation

Limitations

- Requires trained classifier
- Less flexible
- Need separate classifier for each type

 Key Insight: Use classifier gradients to guide generation toward desired conditions

Classifier-Free Guidance

Part 4/7: Sampling



Innovation: No separate classifier needed

🎓 Training Strategy

Randomly drop condition c with probability p (e.g., 10%)

Unconditional

$$\varepsilon_{\theta}(x_t, t, \emptyset)$$

+

Weight

w

×

Difference

$$(\varepsilon_{\theta}(x_t, t, c) - \varepsilon_{\theta}(x_t, t, \emptyset))$$

⚡ Guided Prediction Formula

$$\tilde{\varepsilon} = \varepsilon_{\theta}(x_t, t, \emptyset) + w \cdot (\varepsilon_{\theta}(x_t, t, c) - \varepsilon_{\theta}(x_t, t, \emptyset))$$



Guidance Scale

$w > 1$ increases strength



Trade-off

Higher w = better align, less diverse



Flexibility

Single model, multiple conditions

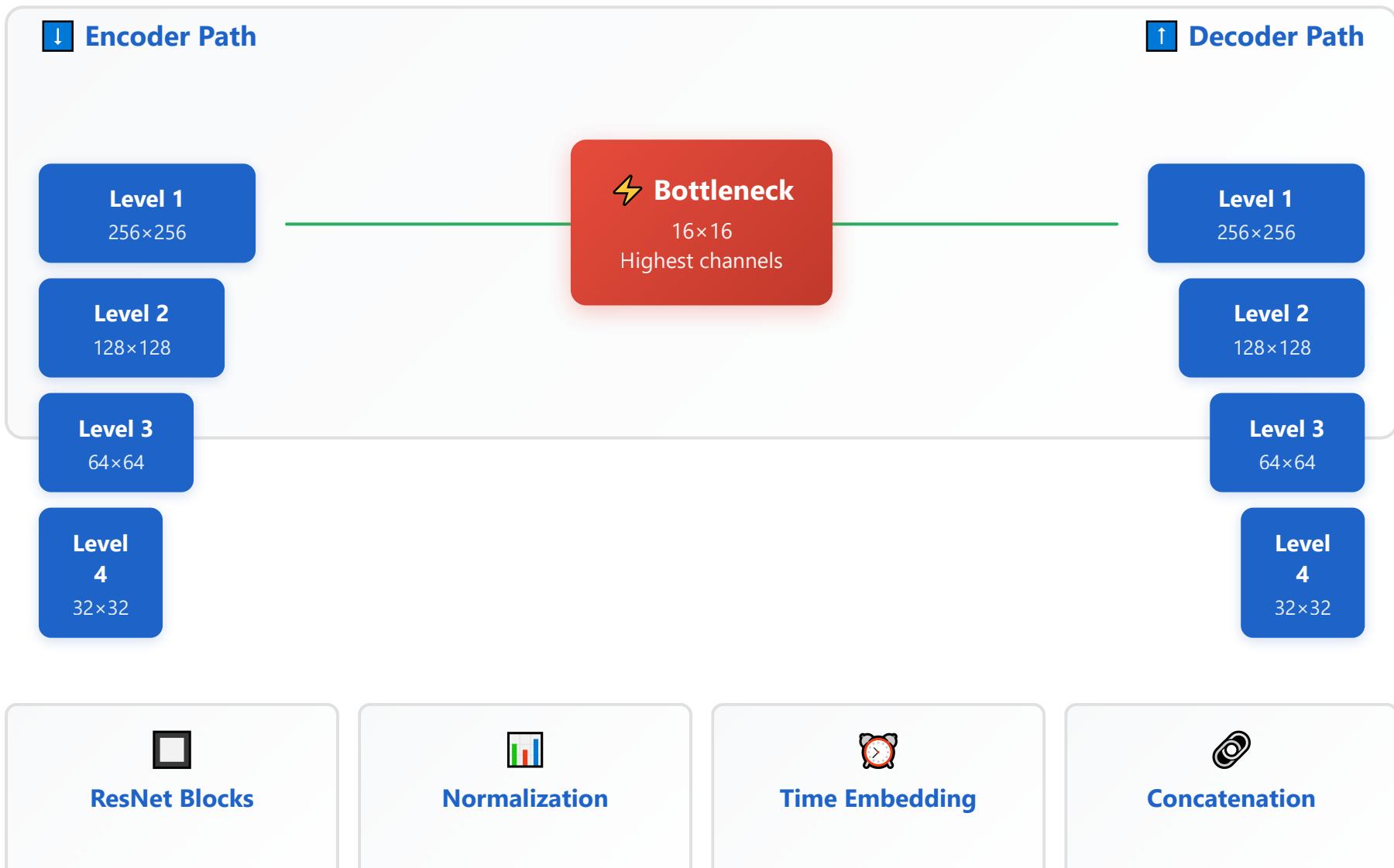
Part 5/7:

Architecture

- 18.** U-Net Structure
- 19.** Attention Mechanism
- 20.** Condition Injection Methods

U-Net Structure

Part 5/7: Architecture



Two conv + residual

Group norm

Adaptive norm

Encoder features

Attention Mechanism

Part 5/7: Architecture



Self-Attention

Q

K

V

Models long-range dependencies in image. Applied at 16×16 , 32×32 resolutions.



Cross-Attention

Text Q



Image K,V

For text conditioning. Text \rightarrow Image attention enables controllable generation.



Key Properties



Multi-Head

Multiple patterns



Spatial Attention

Different locations



Complexity

$O(n^2)$



Mechanism

Standard transformer



Location

Low resolution



Dependencies

Long-range



Impact: Crucial for coherent, high-quality generation

Condition Injection Methods

Part 5/7: Architecture



Time Embedding

Sinusoidal positional encoding + MLP



Class Embedding

Learned embedding for each class



Text Conditioning

CLIP or T5 text encoder → cross-attention



Adaptive Group Norm

Scale and shift from condition embedding



FiLM Layers

Feature-wise Linear Modulation



Concatenation

Directly concat condition to input channels



Cross-Attention

Text tokens attend to image features



Hybrid Approaches

Combine multiple conditioning methods for optimal performance

Categorization by Type



Temporal

Time Embedding



Categorical

Class Embedding



Textual

Text + Cross-Attention



Modern models often combine multiple injection methods for best results

Part 6/7:

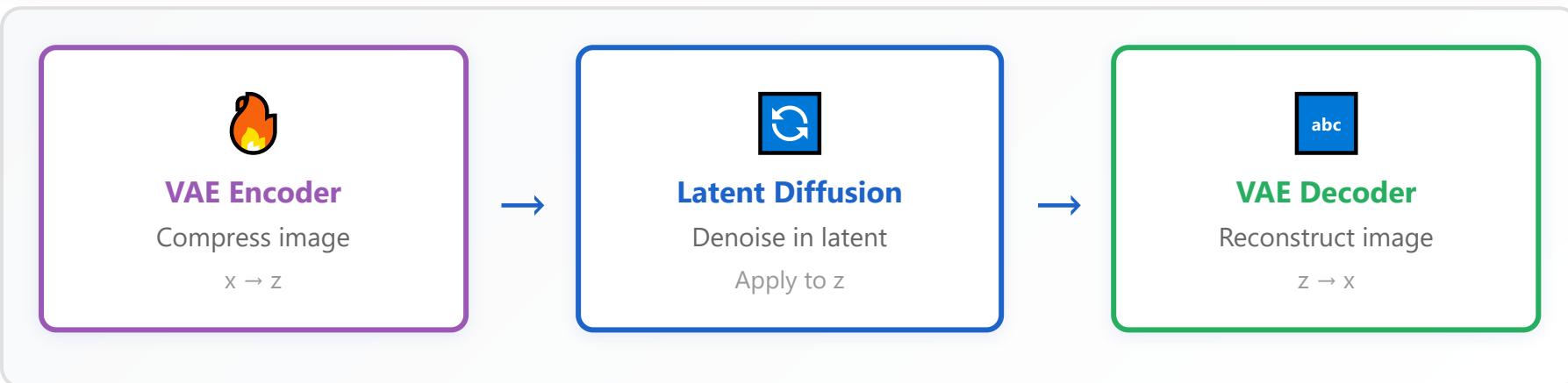
Advanced Techniques

- 21.** Latent Diffusion (Stable Diffusion)
- 22.** Noise Schedule Improvements
- 23.** Other Improvement Techniques

Latent Diffusion (Stable Diffusion)

Part 6/7: Advanced Techniques

⚠ Motivation: Pixel-space diffusion is computationally expensive



Efficiency

64x speedup



Compression

8×8 reduction



Quality

Perceptual loss



Memory

Consumer GPUs



Stable Diffusion: Open-source latent diffusion model

High-resolution generation



Practical Example: 512×512 Image Generation

Pixel-space Diffusion

Input size:

$512 \times 512 \times 3 = 786,432$

Latent Diffusion (8× compression)

Compressed size:

$64 \times 64 \times 4 = 16,384$

VS

Memory per step:

~3 MB

50 steps total:

~150 MB ✗

Memory per step:

~65 KB

50 steps total:

~3.2 MB ✓



Memory Reduction: 48× | Speed Improvement: 64× | Quality: Preserved via perceptual loss



Interactive Demo:

[Diffusion Explainer](#)

Noise Schedule Improvements

Part 6/7: Advanced Techniques

📈 Evolution of Noise Schedules



Original

Linear Schedule

Original DDPM approach

$$\beta_t: 1e-4 \rightarrow 0.02$$


Improved

Cosine Schedule

Smoother transition, better for high-res



Advanced

Learned Schedule

Network learns optimal β_t during training

💡 Key Concepts & Innovations



Signal-to-Noise Ratio

$$\text{SNR}(t) = \bar{\alpha}_t / (1 - \bar{\alpha}_t)$$



V-parameterization

Predict velocity v_t instead of noise



EDM Framework

Exponential noise for better scaling



Continuous Time

Formulate as continuous SDE



Impact: Better sample quality and training stability

Other Improvement Techniques

Part 6/7: Advanced Techniques



Cascaded Diffusion

Low-res generation → super-resolution



Upsampler

Specialized diffusion model for upscaling



Progressive Distillation

Student mimics T-step teacher in T/2 steps



Consistency Models

One-step generation via consistency training



Flow Matching

Alternative to diffusion with optimal transport



Rectified Flow

Straightens probability paths for efficiency



Adversarial Diffusion

Add adversarial loss for sharper samples



Truncation Tricks

Early stopping or partial diffusion for control

📁 Categorization by Purpose



Quality



Speed



Alternative



Control

3 techniques

2 techniques

2 techniques

1 technique

Part 7/7:

Applications and Extensions

24. Applications and Conclusion

Applications and Conclusion

Part 7/7: Applications and Extensions



Text-to-Image

Popular

DALL-E 2, Stable Diffusion, Midjourney, Imagen



Video Generation

Temporal diffusion models



Audio Synthesis

Diffusion for music and speech generation



Medical Imaging

Denoising, reconstruction, synthesis



Image Editing

Inpainting, outpainting, style transfer



3D Generation

NeRF + diffusion (DreamFusion)



Molecular Design

Drug discovery and protein design



Future Directions



Real-time
Generation



Better
Control



Improved
Efficiency

Thank you

Ho-min Park

homin.park@ghent.ac.kr

powersimmani@gmail.com