ML Advanced Topics 2

Diffusion Models for Generative AI

Learning Objectives:

- Understand the mathematical foundations of diffusion processes
- Master the forward and reverse diffusion processes
- Learn DDPM, DDIM, and Score-Based Models
- Explore modern applications: DALL-E 2, Stable Diffusion, video generation

Generative Models Landscape

Three Major Paradigms:

1. **GANs (Generative Adversarial Networks)**
   - Generator vs Discriminator game
   - Fast sampling, but training instability
   - Mode collapse issues
2. **VAEs (Variational Autoencoders)**
   - Encode to latent space, decode to reconstruct
   - Stable training, but blurry outputs
   - Limited sample quality

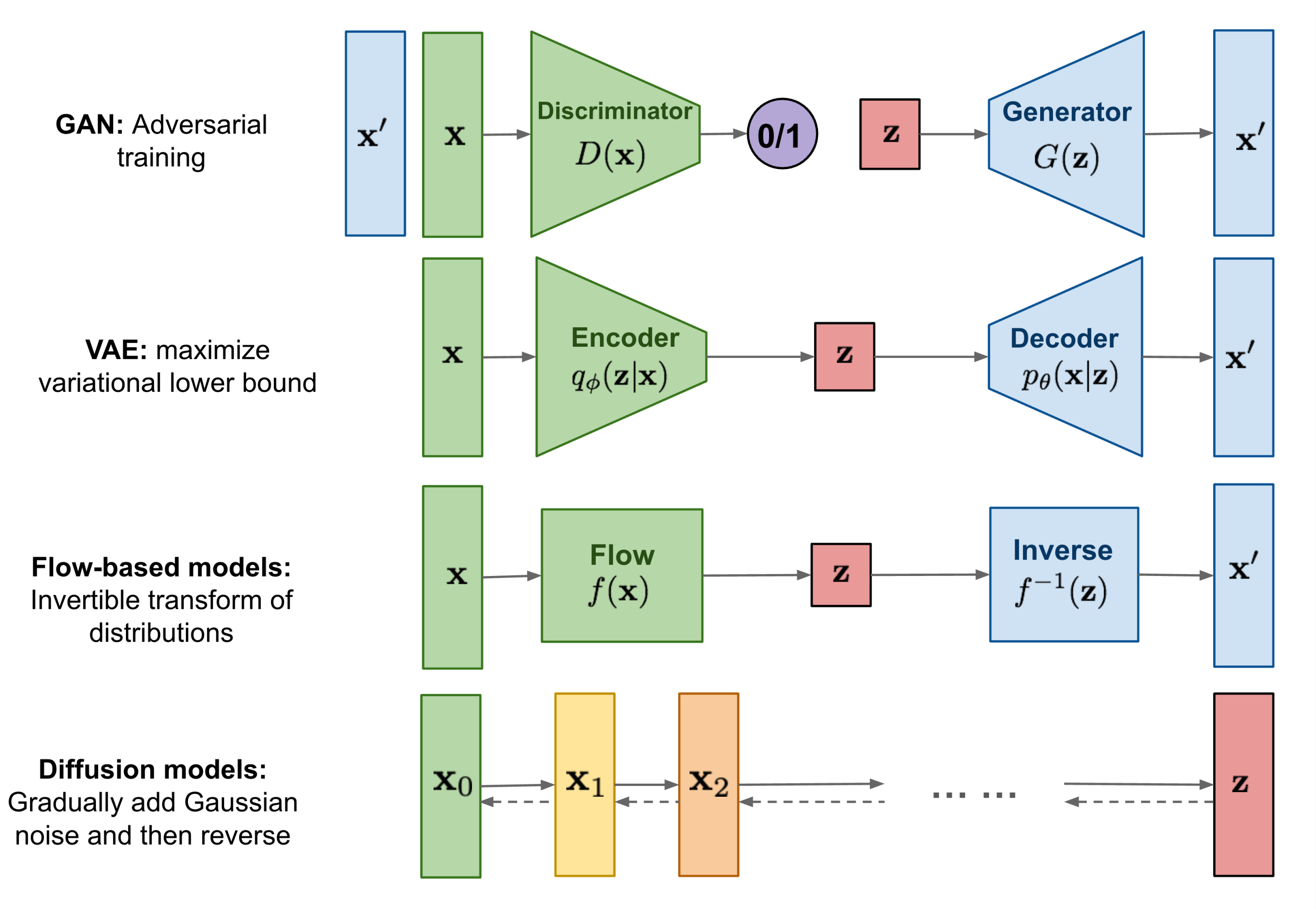
Generative Models Landscape

Three Major Paradigms:

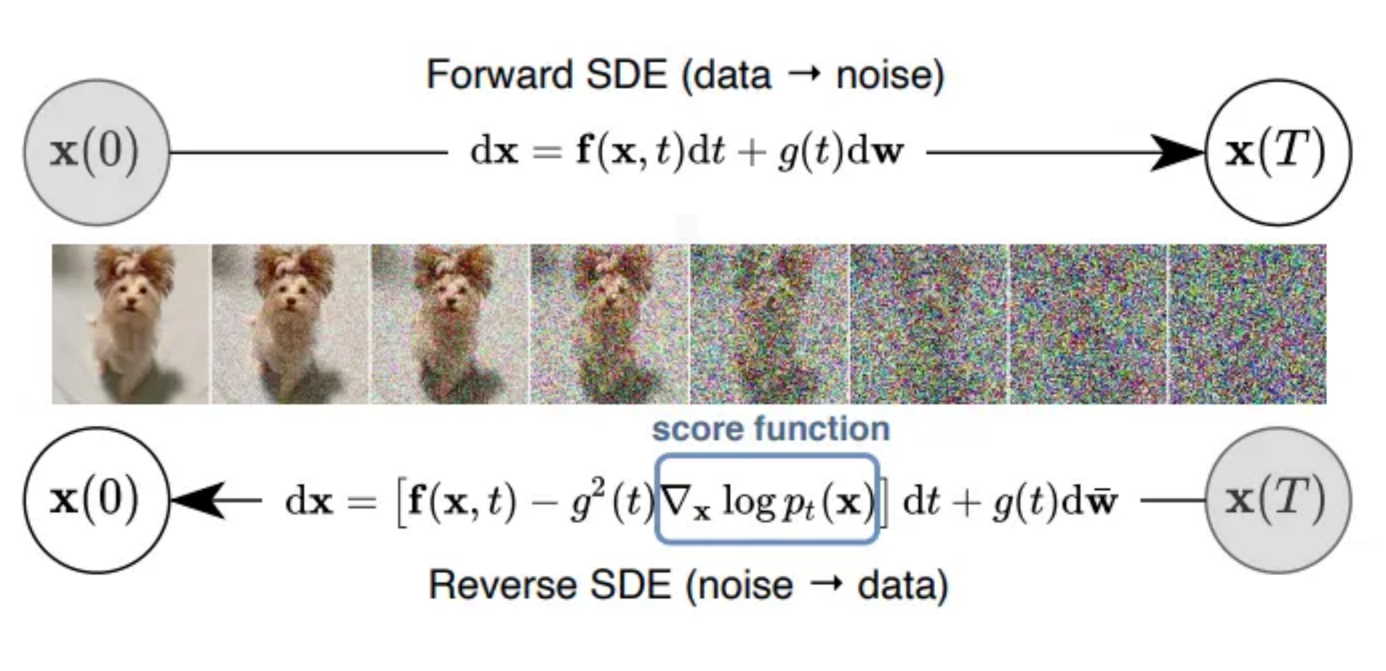
3. Diffusion Models ← Today's Focus

- Gradual denoising process
- State-of-the-art sample quality
- Stable training, but slower sampling

Generative Models Landscape



What Are Diffusion Models?



What Are Diffusion Models?

Core Idea:

Learn to reverse a gradual noising process.

Two Processes:

1. **Forward (Diffusion):** Gradually add noise to data

$$\mathbf{x}_0 \to \mathbf{x}_1 \to \mathbf{x}_2 \to \cdots \to \mathbf{x}_T \approx \mathcal{N}(0, \mathbf{I})$$

2. **Reverse (Denoising):** Learn to remove noise step-by-step

$$\mathbf{x}_T \to \mathbf{x}_{T-1} \to \cdots \to \mathbf{x}_1 \to \mathbf{x}_0$$

Key Insight: If we can reverse the diffusion, we can generate new samples by starting from pure noise!

Forward Diffusion Process

Mathematical Formulation:

Add Gaussian noise at each timestep t:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

where β_t is the variance schedule (small values, typically 10^{-4} to 0.02).

Equivalently:

$$\mathbf{x}_t = \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\boldsymbol{\epsilon}_{t-1}, \quad \boldsymbol{\epsilon}_{t-1} \sim \mathcal{N}(0, \mathbf{I})$$

Nice Property: Can sample $\mathbf{x}_t$ directly from $\mathbf{x}_0$:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

where $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$

Forward Process: Closed Form

Reparameterization:

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$$

Key Properties:

- As $t \to T$, $\bar{\alpha}_t \to 0$
- At $t = T$: $\mathbf{x}_T \approx \mathcal{N}(0, \mathbf{I})$ (pure noise)
- No learnable parameters in forward process!

Forward Process: Closed Form

Variance Schedule Design:

Common choices:

- **Linear:** $\beta_t = \beta_{\min} + \frac{t}{T}(\beta_{\max} - \beta_{\min})$
- **Cosine:** $\bar{\alpha}_t = \frac{f(t)}{f(0)}$, where $f(t) = \cos\left(\frac{t/T+s}{1+s} \cdot \frac{\pi}{2}\right)^2$

Reverse Diffusion Process

Goal: Learn to reverse the forward process.

Reverse Distribution (if known):

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t\mathbf{I})$$

where (by Bayes' rule):

$$\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t$$

$$\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

Problem: We don't know $\mathbf{x}_0$ when sampling!

Solution: Learn to predict $\mathbf{x}_0$ (or the noise $\boldsymbol{\epsilon}$) using a neural network.

Learning the Reverse Process

Parameterize reverse process with neural network:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$$

Variational Lower Bound (ELBO):

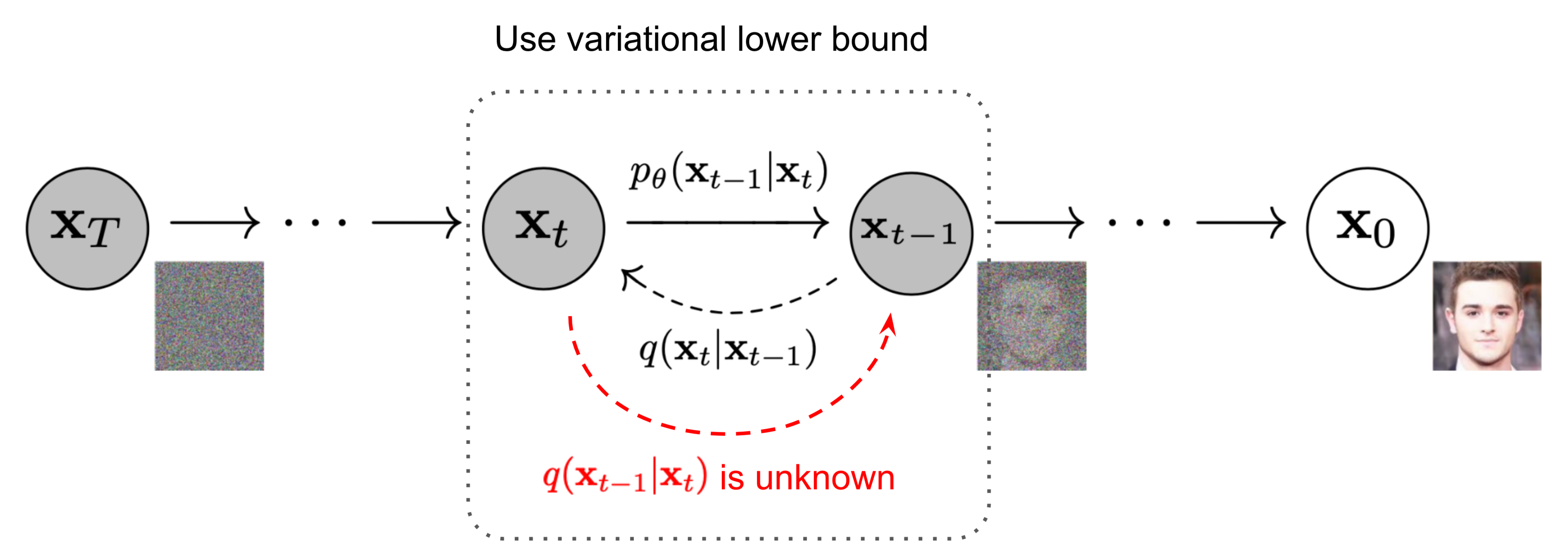
$$\mathbb{E}_{q(\mathbf{x}_0)}\left[\log p_\theta(\mathbf{x}_0)\right] \geq \mathbb{E}_q\left[\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right]$$

Simplified Loss (DDPM, Ho et al. 2020):

$$\mathcal{L}_{\text{simple}} = \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\|^2\right]$$

Interpretation: Train network to predict the noise that was added!

DDPM: Denoising Diffusion Probabilistic Models



DDPM: Denoising Diffusion Probabilistic Models

Algorithm (Ho et al., NeurIPS 2020):

Training:

1. Sample $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ (real data)
2. Sample $t \sim \mathrm{Uniform}(1, T)$
3. Sample $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$
4. Compute $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}$
5. Minimize: $\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\|^2$

DDPM: Denoising Diffusion Probabilistic Models

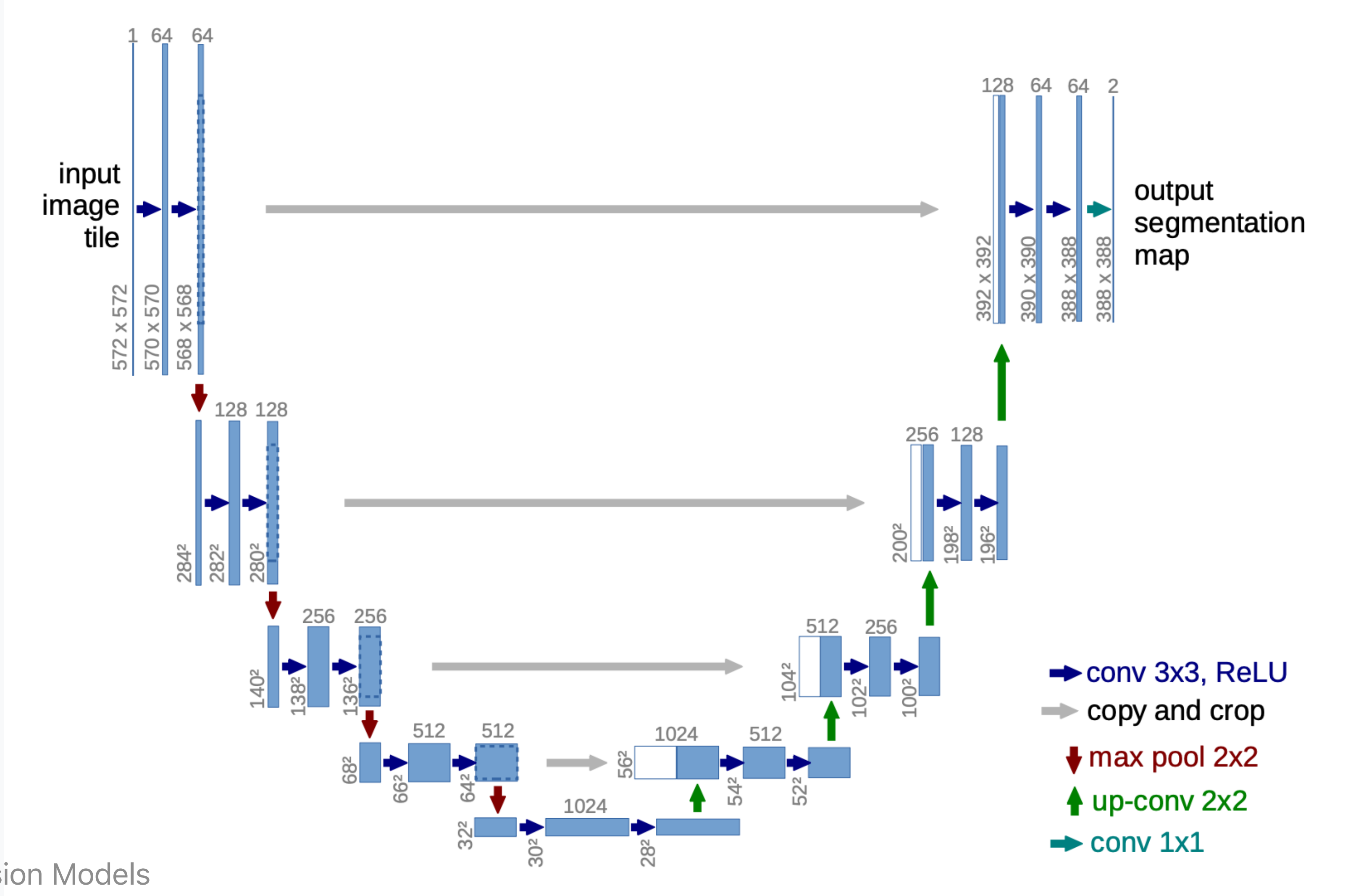
Sampling:

1. Sample $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$
2. For $t = T, T-1, \ldots, 1$:

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sqrt{\beta_t}\mathbf{z}$$

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

Network Architecture: U-Net



Network Architecture: U-Net

Why U-Net?

- Encoder-decoder structure preserves spatial information
- Skip connections for multi-scale features
- Time embedding for timestep conditioning

Time Embedding

Sinusoidal Position Encoding (from Transformers):

$$\mathrm{PE}(t, 2i) = \sin(t/10000^{2i/d})$$
$$\mathrm{PE}(t, 2i+1) = \cos(t/10000^{2i/d})$$

Processing:

1. Create positional encoding for timestep t
2. Pass through 2-layer MLP: $\mathrm{MLP}(\mathrm{PE}(t)) \to \mathbf{e}_t$
3. Inject into U-Net via:
   - **AdaGN (Adaptive Group Norm):** Scale and shift after normalization
   - **Addition:** Add to feature maps
   - **Concatenation:** Concat to spatial features

Time Embedding

Why Important?

Network needs to know how much noise is present (i.e., which timestep) to denoise appropriately!

Attention Mechanisms in Diffusion Models

Self-Attention in U-Net:

At resolution 16×16 or 32×32 (computational cost):

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V}$$

where $\mathbf{Q} = \mathbf{W}_Q\mathbf{h}$, $\mathbf{K} = \mathbf{W}_K\mathbf{h}$, $\mathbf{V} = \mathbf{W}_V\mathbf{h}$

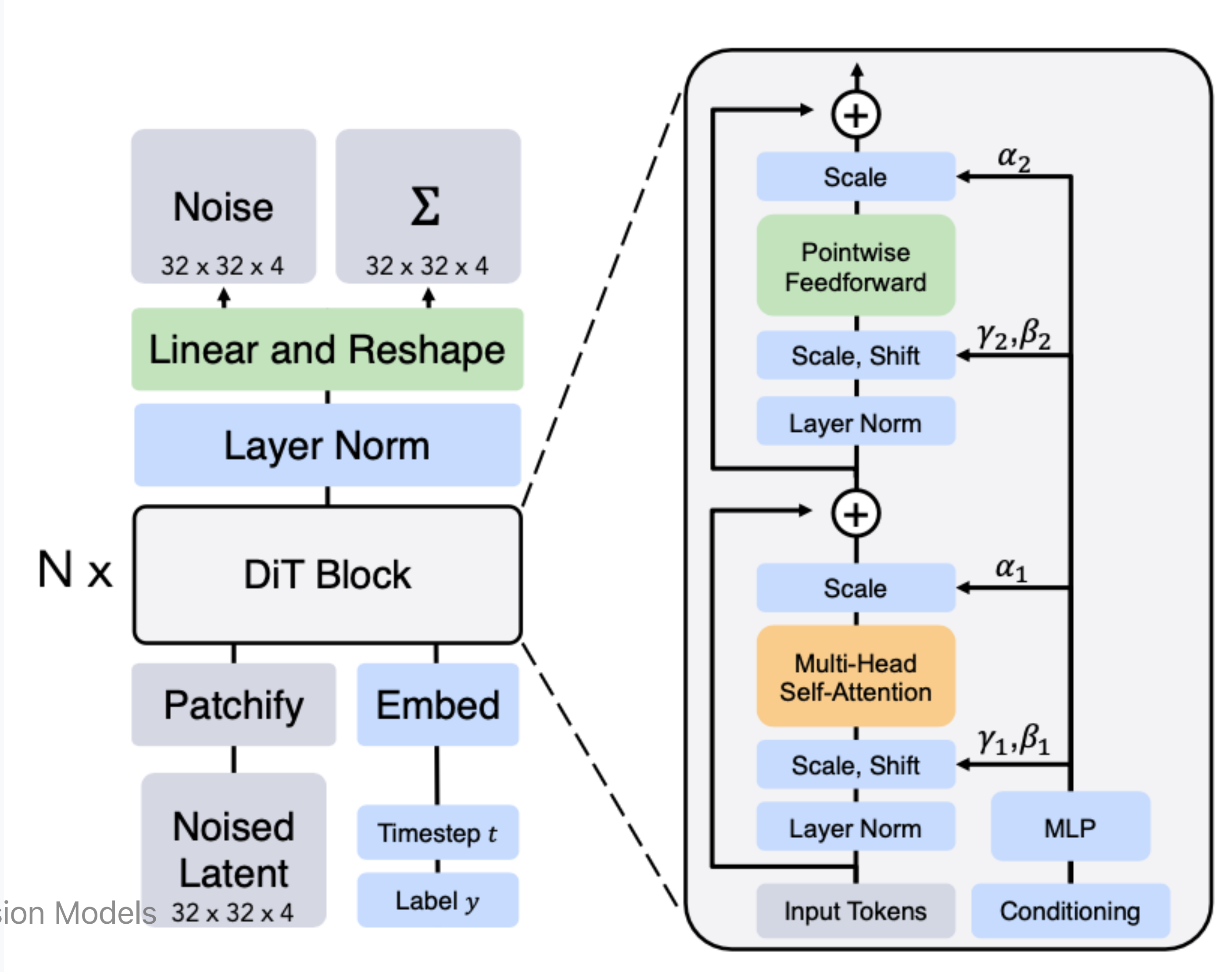
Multi-Head Attention:

Run h attention heads in parallel, concatenate outputs.

Why Attention?

- Captures long-range dependencies, Critical for coherent image generation
- Essential for text-to-image models (cross-attention with text embeddings)

Attention Mechanisms in Diffusion Models



Score-Based Models Connection

Score Function:

$$\nabla_{\mathbf{x}} \log p(\mathbf{x}) = \mathbf{s}_\theta(\mathbf{x})$$

The score points toward higher probability regions.

Score Matching:

$$\mathcal{L}_{\text{score}} = \mathbb{E}_{p(\mathbf{x})}\left[\|\mathbf{s}_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \log p(\mathbf{x})\|^2\right]$$

Denoising Score Matching (Vincent, 2011):

Add noise, then predict score of noisy distribution:

$$\mathcal{L}_{\text{DSM}} = \mathbb{E}_{p(\mathbf{x}),p(\tilde{\mathbf{x}}|\mathbf{x})}\left[\|\mathbf{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log p(\tilde{\mathbf{x}}|\mathbf{x})\|^2\right]$$

Connection to Diffusion:

Predicting noise $\boldsymbol{\epsilon}$ is equivalent to predicting the score!

Score-Based Generative Models (Song & Ermon)

Noise Conditional Score Networks:

Train score network at multiple noise levels:

$$\mathbf{s}_\theta(\mathbf{x}, \sigma)$$

Langevin Dynamics Sampling:

Given score function, sample via:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \epsilon \nabla_{\mathbf{x}} \log p(\mathbf{x}_t) + \sqrt{2\epsilon}\mathbf{z}_t$$

where $\mathbf{z}_t \sim \mathcal{N}(0, \mathbf{I})$

Score-Based Generative Models (Song & Ermon)

Unified Framework (Song et al., ICLR 2021):

DDPMs and Score-Based Models are equivalent under continuous-time formulation (SDEs)!

SDE Formulation:

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + g(t)d\mathbf{w}$$

DDIM: Faster Sampling

Problem with DDPM:

Sampling requires T steps (e.g., $T = 1000$) → slow!

DDIM (Song et al., ICLR 2021):

Denoising Diffusion Implicit Models - non-Markovian process.

Key Idea:

Define deterministic sampling process that still inverts the same forward process.

DDIM Sampling:

$$\mathbf{x}_{t-1} = \sqrt{\bar{\alpha}_{t-1}} \underbrace{\left(\frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)}{\sqrt{\bar{\alpha}_t}} \right)}_{\text{predicted } \mathbf{x}_0} + \underbrace{\sqrt{1-\bar{\alpha}_{t-1} - \sigma_t^2} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)}_{\text{direction toward } \mathbf{x}_t}$$

DDIM: Faster Sampling

Advantages:

- Can skip timesteps: $T = 1000 \to 50$ steps (20× faster!)
- Deterministic when $\sigma_t = 0$
- Same quality as DDPM with fewer steps

Classifier Guidance

Goal: Generate samples conditioned on class label y.

Bayes Rule:

$$\nabla_{\mathbf{x}} \log p(\mathbf{x}|y) = \nabla_{\mathbf{x}} \log p(\mathbf{x}) + \nabla_{\mathbf{x}} \log p(y|\mathbf{x})$$

Guided Sampling:

$$\boldsymbol{\epsilon}_{\text{guided}} = \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) - w \cdot \sqrt{1 - \bar{\alpha}_t} \nabla_{\mathbf{x}_t} \log p_\phi(y|\mathbf{x}_t)$$

where w is guidance weight, $p_\phi(y|\mathbf{x}_t)$ is a pre-trained classifier.

Classifier Guidance

Effect:

- $w > 1$: Stronger conditioning (more class-consistent, less diverse)
- $w = 1$: Standard conditional generation
- $w < 1$: Weaker conditioning (more diverse, less consistent)

Problem: Requires separate classifier training.

Classifier-Free Guidance

Idea: Train a single model that handles both conditional and unconditional generation.

Training:

Randomly drop conditioning y with probability p (e.g., 10%):

$$\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, y), \quad \text{or} \quad \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, \emptyset)$$

Sampling:

Interpolate between conditional and unconditional predictions:

$$\tilde{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t, t, y) = \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, \emptyset) + w \cdot (\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, y) - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, \emptyset))$$

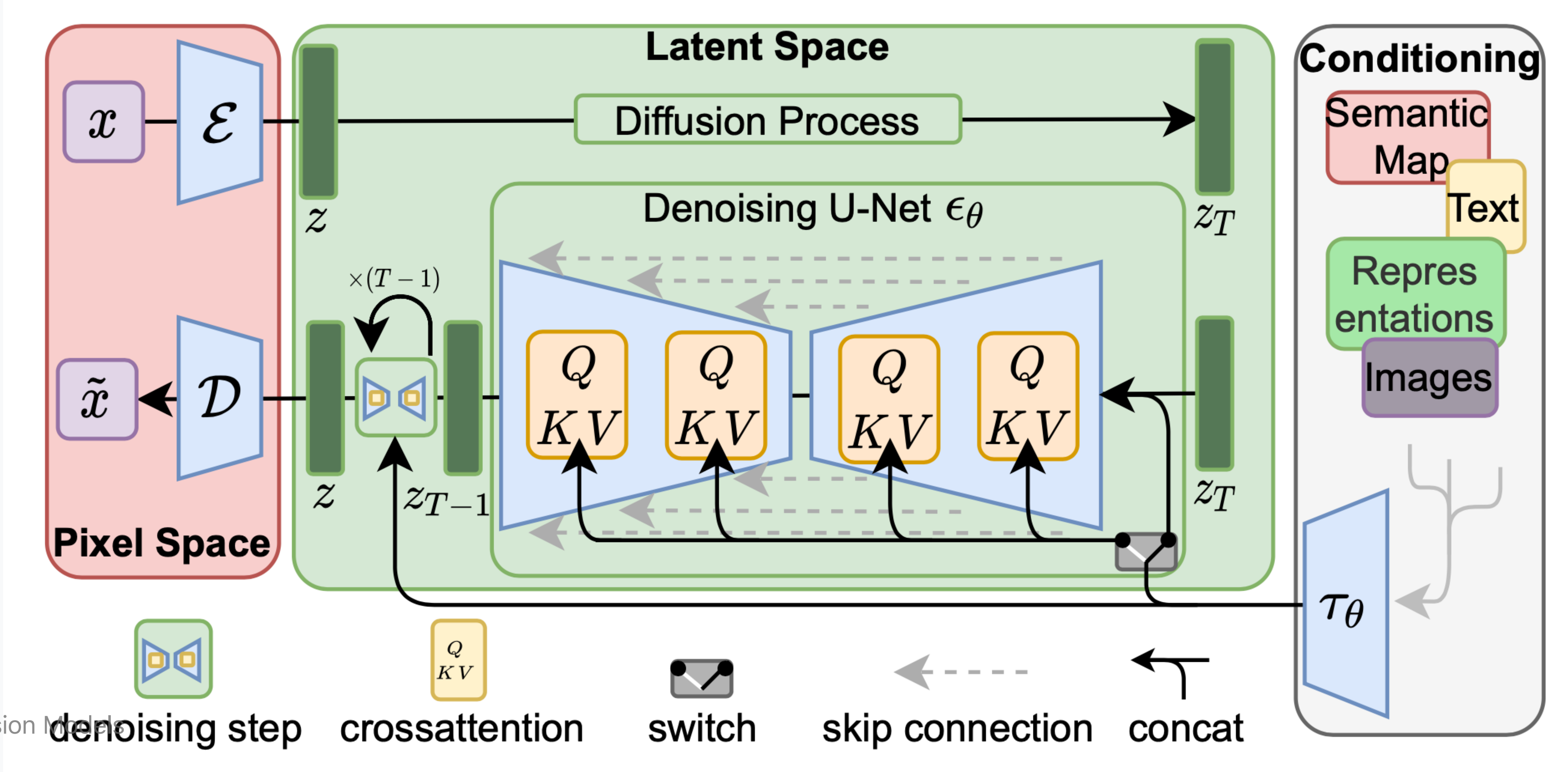
Classifier-Free Guidance

Advantages:

- No separate classifier needed
- Better sample quality than classifier guidance
- Widely used in DALL-E 2, Stable Diffusion, Imagen

Typical values: $w \in [5, 15]$

Latent Diffusion Models (Stable Diffusion)



Latent Diffusion Models (Stable Diffusion)

Problem: High-resolution images (e.g., 512×512) → expensive!

Solution: Operate in latent space of a pre-trained autoencoder.

Architecture:

1. **Encoder:** $\mathcal{E} : \mathbb{R}^{H \times W \times 3} \to \mathbb{R}^{h \times w \times c}$
   - Downsample: $512 \times 512 \to 64 \times 64$ (8× compression)
2. **Diffusion:** Apply diffusion in latent space $\mathbf{z}$
   - Much cheaper: $(64 \times 64)^2 / (512 \times 512)^2 = 1/64$ pixels!
3. **Decoder:** $\mathcal{D} : \mathbb{R}^{h \times w \times c} \to \mathbb{R}^{H \times W \times 3}$
   - Upsample back to image space

Latent Diffusion Models (Stable Diffusion)

Training:

$$\mathcal{L}_{\text{LDM}} = \mathbb{E}_{\mathcal{E}(\mathbf{x}),\boldsymbol{\epsilon},t}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{z}_t, t, \tau_\theta(y))\right\|^2\right]$$

where $\tau_\theta(y)$ is text embedding (e.g., CLIP).

Cross-Attention for Text Conditioning

Text Encoder:

- CLIP (Contrastive Language-Image Pre-training)
- T5, BERT, or similar transformer
- Output: Text embeddings $\tau_\theta(y) \in \mathbb{R}^{L \times d}$ (sequence of tokens)

Cross-Attention for Text Conditioning

Cross-Attention in U-Net:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V}$$

where:

- $\mathbf{Q} = \mathbf{W}_Q\mathbf{h}$ (query from image features)
- $\mathbf{K} = \mathbf{W}_K\tau_\theta(y)$ (key from text)
- $\mathbf{V} = \mathbf{W}_V\tau_\theta(y)$ (value from text)

Effect:

Each spatial location in image attends to relevant parts of text description!

Modern Diffusion Models Landscape

Text-to-Image:

- **DALL-E 2 (OpenAI, 2022):** CLIP embeddings + diffusion
- **Stable Diffusion (Stability AI, 2022):** Open-source latent diffusion
- **Imagen (Google, 2022):** Large language model embeddings + cascaded diffusion
- **Midjourney:** Commercial, artistic quality

Video Generation:

- **Imagen Video (Google, 2022):** Spatiotemporal U-Net
- **Make-A-Video (Meta, 2022):** Text-to-video with unsupervised learning
- **Runway Gen-2:** Commercial video generation

Modern Diffusion Models Landscape

3D Generation:

- **DreamFusion (Google, 2022):** Text-to-3D via NeRF + diffusion
- **Point-E (OpenAI, 2022):** Text-to-3D point clouds

Audio:

- **AudioLM (Google, 2022):** Audio generation
- **Riffusion:** Music generation

DALL-E 2 Architecture

Two-Stage Process:

Stage 1: Text to CLIP Image Embedding

- Text $\rightarrow$ CLIP text embedding
- **Prior:** Diffusion model or autoregressive model
- Generates CLIP image embedding from text embedding

$$\mathbf{z}_{\text{img}} \sim p(\mathbf{z}_{\text{img}} | \mathbf{z}_{\text{text}})$$

Stage 2: CLIP Embedding to Image

- **Decoder:** Diffusion model conditioned on CLIP embedding
- Upsampling diffusion models (64×64 → 256×256 → 1024×1024)

Key Innovation: Operating in CLIP embedding space provides semantic guidance!

Evaluation Metrics

Quantitative Metrics:

1. FID (Fréchet Inception Distance):

$$\text{FID} = \|\mu_r - \mu_g\|^2 + \text{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2})$$

- Lower is better (measures distribution similarity)
- Computed on Inception features

2. IS (Inception Score):

$$\text{IS} = \exp(\mathbb{E}_x[\text{KL}(p(y|x)\|p(y))])$$

- Higher is better (quality + diversity)

Evaluation Metrics

3. **Precision & Recall:**
    - **Precision:** Generated samples are realistic
    - **Recall:** Generated samples cover real distribution
4. **CLIP Score (for text-to-image):**
    - Cosine similarity between CLIP embeddings of generated image and text

Inpainting with Diffusion Models

Goal: Fill in missing regions of an image.

Method 1: Repaint (Lugmayr et al., 2022)

At each denoising step:

1. Denoise entire image: $\mathbf{x}'_{t-1} \sim p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$
2. Replace known regions with forward diffusion from original:

$$\mathbf{x}_{t-1} = \mathbf{m} \odot \mathbf{x}_{t-1}^{\text{known}} + (1 - \mathbf{m}) \odot \mathbf{x}'_{t-1}$$

where $\mathbf{m}$ is binary mask (1 = known, 0 = unknown).

Method 2: Train with Random Masks

During training, randomly mask parts of input and condition on mask:

$$\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, \mathbf{m}, \mathbf{x}_{\text{masked}})$$

Inpainting with Diffusion Models

Applications:

- Object removal
- Image extension (outpainting)
- Super-resolution

Image Editing with Diffusion

1. **SDEdit (Meng et al., ICLR 2022):**
   - Add noise to real image: $\mathbf{x}_0 \to \mathbf{x}_t$
   - Denoise with guidance: $\mathbf{x}_t \to \mathbf{x}_0'$
   - Preserves structure, changes content
2. **Prompt-to-Prompt (Hertz et al., 2022):**
   - Edit text prompt
   - Use cross-attention from original generation
   - Fine-grained control over edits
3. **DreamBooth (Ruiz et al., 2022):**
   - Fine-tune diffusion model on 3-5 images

Super-Resolution with Diffusion

Cascaded Diffusion (Ho et al., 2022):

Generate image at increasing resolutions:

$$64 \times 64 \to 256 \times 256 \to 1024 \times 1024$$

SR3 (Saharia et al., 2022):

Conditioning on low-resolution image:

$$p(\mathbf{x}_{HR} | \mathbf{x}_{LR})$$

Implementation:

- Concatenate low-res image (upsampled) with noisy high-res
- Train diffusion model to denoise
- At inference: condition on any low-res input

Super-Resolution with Diffusion

Applications:

- Photo enhancement
- Medical imaging
- Satellite imagery

Graph-Based Generative Models

Why Graphs?

- Molecules, proteins, social networks, knowledge graphs
- Non-Euclidean structure (variable size, permutation invariant)
- Traditional diffusion assumes fixed grid → need adaptation!

Graph Representation:

$$\mathcal{G} = (\mathbf{X}, \mathbf{A})$$

- $\mathbf{X} \in \mathbb{R}^{N \times d}$: Node features (atom types, charges)
- $\mathbf{A} \in \{0, 1\}^{N \times N}$: Adjacency matrix (bonds)

Graph-Based Generative Models

Key Challenges:

1. Discrete structure (edges exist or don't)
2. Variable graph sizes
3. Permutation invariance/equivariance
4. Validity constraints (chemical rules)

Diffusion on Graphs: Core Ideas

Two Approaches:

1. Continuous Relaxation:

- Treat discrete edges as continuous values
- Add Gaussian noise to node features $\mathbf{X}$ and edge features
- Discretize at final step

2. Discrete Diffusion:

- Define forward process over discrete states
- Categorical noise instead of Gaussian
- Absorbing state or uniform diffusion

Diffusion on Graphs: Core Ideas

Forward Process (Continuous):

$$q(\mathbf{X}_t|\mathbf{X}_0) = \mathcal{N}(\mathbf{X}_t; \sqrt{\bar{\alpha}_t}\mathbf{X}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

$$q(\mathbf{A}_t|\mathbf{A}_0) = \text{discretize}(\sqrt{\bar{\alpha}_t}\mathbf{A}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon})$$

Graph Neural Networks for Denoising

Equivariant Denoising Network:

Must respect permutation symmetry of graphs!

Message Passing Architecture:

$$\mathbf{m}_{ij}^{(l)} = \phi_e(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}, \mathbf{e}_{ij})$$

$$\mathbf{h}_i^{(l+1)} = \phi_h\left(\mathbf{h}_i^{(l)}, \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij}^{(l)}\right)$$

Graph Neural Networks for Denoising

Time Conditioning:

- Add timestep embedding to node features
- Similar to U-Net time embedding

$$\mathbf{h}_i^{(0)} = [\mathbf{x}_i; \mathrm{MLP}(\mathrm{PE}(t))]$$

Output:

- Predict noise on node features: $\boldsymbol{\epsilon}_\theta^X(\mathbf{X}_t, \mathbf{A}_t, t)$
- Predict noise on edge features: $\boldsymbol{\epsilon}_\theta^E(\mathbf{X}_t, \mathbf{A}_t, t)$

DiGress: Discrete Denoising Diffusion for Graphs

Discrete Diffusion (Vignac et al., ICLR 2023):

Forward Process (Categorical):

$$q(\mathbf{X}_t|\mathbf{X}_{t-1}) = \mathrm{Cat}(\mathbf{X}_{t-1}\mathbf{Q}_t^X)$$
$$q(\mathbf{A}_t|\mathbf{A}_{t-1}) = \mathrm{Cat}(\mathbf{A}_{t-1}\mathbf{Q}_t^E)$$

where $\mathbf{Q}_t$ are transition matrices (e.g., toward uniform or absorbing state).

Closed Form:

$$q(\mathbf{X}_t|\mathbf{X}_0) = \mathrm{Cat}(\mathbf{X}_0\bar{\mathbf{Q}}_t^X), \quad \bar{\mathbf{Q}}_t = \prod_{s=1}^{t}\mathbf{Q}_s$$

DiGress: Discrete Denoising Diffusion for Graphs

Reverse Process:

$$p_\theta(\mathbf{X}_{t-1}|\mathbf{X}_t, \mathbf{A}_t) \propto q(\mathbf{X}_t|\mathbf{X}_{t-1}) \cdot p_\theta(\mathbf{X}_0|\mathbf{X}_t, \mathbf{A}_t)$$

Network predicts clean graph $(\hat{\mathbf{X}}_0, \hat{\mathbf{A}}_0)$ directly!

E(3) Equivariant Diffusion for Molecules

3D Molecular Generation:

Molecules have 3D coordinates → need SE(3) or E(3) equivariance!

E(3) Equivariant Graph Neural Networks (EGNN):

$$\mathbf{h}_i^{(l+1)} = \phi_h(\mathbf{h}_i^{(l)}, \sum_j \phi_m(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}, \|\mathbf{r}_i - \mathbf{r}_j\|^2))$$

$$\mathbf{r}_i^{(l+1)} = \mathbf{r}_i^{(l)} + \sum_j (\mathbf{r}_i - \mathbf{r}_j)\phi_r(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}, \|\mathbf{r}_i - \mathbf{r}_j\|^2)$$

E(3) Equivariant Diffusion for Molecules

EDM (Hoogeboom et al., ICML 2022):

- Diffusion on both atom types (discrete) and coordinates (continuous)
- Coordinates: zero center-of-mass constraint
- Preserves rotation/translation invariance

Applications:

- Drug discovery
- Materials design
- Protein structure generation

Conditional Graph Generation

Property-Guided Generation:

Generate molecules with target properties (e.g., binding affinity, drug-likeness).

Classifier-Free Guidance for Graphs:

$$\tilde{\boldsymbol{\epsilon}}_\theta = \boldsymbol{\epsilon}_\theta(\mathcal{G}_t, t, \emptyset) + w \cdot (\boldsymbol{\epsilon}_\theta(\mathcal{G}_t, t, c) - \boldsymbol{\epsilon}_\theta(\mathcal{G}_t, t, \emptyset))$$

Context Conditioning:

- Target molecular properties (LogP, QED, binding score)
- Scaffold constraints (generate around fixed substructure)
- Protein pocket (generate ligand conditioned on binding site)

Conditional Graph Generation

Applications:

- **De novo drug design:** Generate molecules binding to target protein
- **Lead optimization:** Modify existing molecules to improve properties
- **Linker design:** Connect molecular fragments

Graph Diffusion Applications

1. **Molecule Generation:**
   - **GDSS (Jo et al., ICML 2022):** Score-based on adjacency + features
   - **EDM (Hoogeboom et al., ICML 2022):** E(3) equivariant for 3D
   - **DiGress (Vignac et al., ICLR 2023):** Discrete categorical diffusion
2. **Protein Structure:**
   - **RFDiffusion (Watson et al., Nature 2023):** Protein backbone generation
   - **Chroma (Ingraham et al., Nature 2023):** Programmable protein design
   - **FrameDiff (Yim et al., ICLR 2024):** SE(3) diffusion on frames

Graph Diffusion Applications

3. Materials Science:

- **CDVAE (Xie et al., ICLR 2022):** Crystal structure generation
- **DiffCSP (Jiao et al., ICLR 2024):** Crystal structure prediction

4. Combinatorial Optimization:

- **DIFUSCO (Sun et al., NeurIPS 2023):** TSP, graph coloring
- Graph diffusion for routing problems

Summary: Graph Diffusion Models

Aspect	Image Diffusion	Graph Diffusion
Data	Fixed grid	Variable structure
Noise	Gaussian	Gaussian + Categorical
Network	U-Net	GNN / Transformer
Symmetry	Translation	Permutation, E(3)
Output	Continuous	Often discrete

Summary: Graph Diffusion Models

Key Innovations:

1. **Discrete diffusion:** Handle categorical node/edge types
2. **Equivariant networks:** Respect graph/geometric symmetries
3. **Validity constraints:** Enforce chemical/structural rules
4. **Conditional generation:** Property-guided design

Future Directions:

- Faster sampling for large graphs
- Multi-modal conditioning (text + structure)
- Scaling to larger molecular systems

References

@article{weng2021diffusion,

title = "What are diffusion models?",

author = "Weng, Lilian",

journal = "lilianweng.github.io",

year = "2021",

month = "Jul",

url = "https://lilianweng.github.io/posts/2021-07-11-diffusion-models/"

}