Diffusion Models & Inpainting

DDP - DDIM - DPS - CFG

Angela Saade - Aurélien Daudin - Baptiste Arnold
Khaled Mili - Maxime Ruff - Pierre Schweitzer

EPITA

January 26, 2026

Outline

DDPM - Denoising Diffusion Probabilistic Models

The DDPM framework consists of two opposing Markov chains.

1. The Forward Process (q):

- A fixed chain that gradually adds Gaussian noise according to a variance schedule β_t.
- Transition kernel:

$$q(x_t|x_{t-1}) := \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

2. The Reverse Process (p_θ):

- A learned Markov chain with Gaussian transitions.
- Parameterized by a neural network to reverse the noise:

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

DDPM - Efficient Forward Sampling

A key property allows us to sample x_t at any timestep t directly from x_0 without iterating.

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

Marginal at timestep t

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t} x_0, (1 - \bar{\alpha}_t) I)$$

Practical Implication: We can generate noisy training samples on the fly:

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

This makes training efficient by optimizing random terms of the variational bound.

DDPM - Simplified Training Objective (L_{simple})

Instead of predicting the mean $\tilde{\mu}_t$, the network ϵ_θ is trained to predict the **noise** ϵ added to the image.

The DDPM paper proposes a simplified weighted variational bound:

Loss Function

$$L_{simple}(\theta) := \mathbb{E}_{t,x_0,\epsilon}\left[\left\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\right\|^2\right]$$

Key Insights:

- It resembles denoising score matching over multiple noise scales.
- It down-weights loss terms at small t to focus on difficult denoising tasks.

Sampling with Langevin Dynamics

To generate data, we start from pure noise $x_T \sim \mathcal{N}(0, I)$ and iterate backwards.

Sampling Step

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

- $\epsilon_\theta(x_t, t)$ estimates the noise to be removed.
- $z \sim \mathcal{N}(0, I)$ adds stochasticity (Langevin dynamics).
- This stochastic term prevents the process from becoming deterministic and collapsing to the mean.

DDPM Limitations

The Problem with DDPM:

- The generative process approximates the reverse of a diffusion process.
- Consequence: Sampling is extremely slow. Generating a single image requires running the neural network 1000 times .

The DDIM Insight:

- The training objective L_{simple} only depends on the marginals $q(x_t|x_0)$, not the specific joint trajectory.
- We can define a **non-Markovian forward process** that leads to the exact same marginals but allows for a different, deterministic reverse process .

Denoising Diffusion Implicit Models (DDIM)

DDIM generalizes the sampling equation.

New Update Equation (Eq. 12 in paper): To go from x_t to x_{t-1}, we predict x_0 (denoted as $f_\theta^{(t)}(x_t)$) and then interpolate:

Generic DDIM Sampling Step

$$x_{t-1} = \underbrace{\sqrt{\bar{\alpha}_{t-1}} \left(\frac{x_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(x_t)}{\sqrt{\bar{\alpha}_t}} \right)}_{\text{Denoised } x_0 \text{ prediction}} + \underbrace{\sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2} \cdot \epsilon_\theta(x_t)}_{\text{Direction pointing to } x_t} + \underbrace{\sigma_t \epsilon}_{\text{Random noise}}$$

The DDIM Case ($\sigma_t = 0$):

- By setting $\sigma_t = 0$, the random noise term vanishes.
- The process becomes **deterministic** (Implicit Model).
- We can use the **same pre-trained model** ϵ_θ as DDPM!.

DDIM - Visuals

$$\boldsymbol{x}_{t-1} = \underbrace{\sqrt{\alpha_{t-1}}\left(\frac{\boldsymbol{x}_t - \sqrt{1-\alpha_t}\,\epsilon_\theta^{(t)}(\boldsymbol{x}_t)}{\sqrt{\alpha_t}}\right)}_{\text{“predicted } \boldsymbol{x}_0\text{”}} + \boxed{\underbrace{\sqrt{1-\alpha_{t-1}-\sigma_t^2}\cdot\epsilon_\theta^{(t)}(\boldsymbol{x}_t)}_{\text{“direction pointing to } \boldsymbol{x}_t\text{”}} + \underbrace{\sigma_t\boldsymbol{\epsilon}_t}_{\substack{\text{random}\\ \text{noise}}}}$$

Step 0 | Step 181 | Step 381 | Step 581 | Step 781 | Step 999

DDPM vs. DDIM: Conclusion

Despite using the **same neural network** trained with the **same loss function**, their inference behaviors differ significantly.

Feature	**DDPM**	**DDIM**
Reverse Process	Stochastic (Markovian).	Deterministic (Non-Markovian).
Speed	Slow (requires $\sim$1000 steps).	Fast (high quality in 50-100 steps).
Latent Space (x_T)	No strong link to x_0 (stochastic mapping).	Fixed encoding of x_0 (allows interpolation and reconstruction).

Classifier Guidance (CG): Mathematical Framework

Goal: Modify conditional score by incorporating signal from auxiliary classifier

Starting point (standard conditional score):

$$\epsilon_\theta(x_t, c) \approx -\sigma_t \nabla_{x_t} \log p(x_t \mid c) \tag{1}$$

Add classifier gradient with weight w:

$$\boxed{\tilde{\epsilon}(x_t, c) = \epsilon_\theta(x_t, c) - w\sigma_t \nabla_{x_t} \log p_\phi(c \mid x_t)} \tag{2}$$

Implicit distribution: Sampling with this modified score approximates

$$\tilde{p}_\theta(x_t \mid c) \propto p_\theta(x_t \mid c) \cdot [p_\phi(c \mid x_t)]^w \tag{3}$$

Intuition:

- Base term: $\epsilon_\theta(x_t, c)$ predicts natural denoising direction
- Correction term: $-w\sigma_t \nabla \log p_\phi$ pushes toward regions where classifier is confident
- Weight w controls strength of this correction

Classifier Guidance: Complete Mathematical Derivation

Step 1: Bayes' rule for conditional probability

$$\nabla_{x_t} \log p(x_t \mid c) = \nabla_{x_t} \log p(x_t) + \nabla_{x_t} \log p(c \mid x_t) \tag{4}$$

Step 2: Scale both sides by $(1+w)$

$$\begin{aligned}(1+w)\nabla_{x_t} \log p(x_t \mid c) &= (1+w)\nabla_{x_t} \log p(x_t) \qquad (5)\\ &\quad + (1+w)\nabla_{x_t} \log p(c \mid x_t) \qquad (6)\end{aligned}$$

Step 3: Expand and rearrange

$$\begin{aligned}&= \nabla_{x_t} \log p(x_t) + w\nabla_{x_t} \log p(x_t) + \nabla_{x_t} \log p(x_t \mid c) \qquad (7)\\ &\quad + w\nabla_{x_t} \log p(c \mid x_t) \qquad (8)\end{aligned}$$

Step 4: Convert to noise predictors using $\epsilon \approx -\sigma\nabla \log p$

Result:

$$\boxed{\tilde{\epsilon}(x_t, c) = \epsilon_\theta(x_t, c) - w\sigma_t \nabla_{x_t} \log p_\phi(c \mid x_t)} \tag{9}$$

Classifier Guidance: Inference Algorithm

Algorithm 1 Classifier Guidance Sampling

Require: condition c, guidance strength w, timesteps $\{t_1, \ldots, t_T\}$

1: **Initialize:** $x_T \sim \mathcal{N}(0, I)$
2: **for** $t = T$ to 1 **do**
3: $\quad \epsilon_{\text{model}} \leftarrow \epsilon_\theta(x_t, t, c)$ // Diffusion forward
4: $\quad$ Compute $\nabla_{x_t} \log p_\phi(c \mid x_t)$ // Classifier forward + backward
5: $\quad \tilde{\epsilon} \leftarrow \epsilon_{\text{model}} - w\sigma_t \nabla_{x_t} \log p_\phi(c \mid x_t)$ // Guidance
6: $\quad x_{t-1} \sim p_\theta(x_{t-1} \mid x_t, c, \tilde{\epsilon})$ // Update step
7: **end for**
8: **return** $x_0 = x_1$

Computational requirements per timestep:

- ✓ 1 forward pass: diffusion model $\epsilon_\theta(x_t, t, c)$
- ✓ 1 forward pass: classifier $p_\phi(c \mid x_t)$
- ✓ 1 backward pass: classifier gradient $\nabla_{x_t} \log p_\phi(c \mid x_t)$
- ✓ 1 linear combination (negligible)

Classifier-Free Guidance (CFG): Key Innovation

Central idea: Use implicit classifier derived from the generative model itself

Implicit classifier (by Bayes' rule):
By Bayes' theorem, we can express the class probability as:

$$p_i(c \mid x_t) = \frac{p(x_t \mid c)\, p(c)}{p(x_t)} \propto \frac{p(x_t \mid c)}{p(x_t)} \tag{10}$$

Classifier-Free Guidance: Complete Derivation

Step 2: Gradient of log (score of implicit classifier)

$$\nabla_{x_t} \log p_i(c \mid x_t) = \nabla_{x_t} \log p(x_t \mid c) - \nabla_{x_t} \log p(x_t) \tag{11}$$

Step 3: Apply score-noise relationship

$$= -\frac{1}{\sigma_t} [\epsilon_\theta(x_t, c) - \epsilon_\theta(x_t)] \tag{12}$$

Step 4: Substitute into CG formula

$$\tilde{\epsilon}(x_t, c) = \epsilon_\theta(x_t, c) - w\sigma_t \cdot \left(-\frac{1}{\sigma_t}\right) [\epsilon_\theta(x_t, c) - \epsilon_\theta(x_t)] \tag{13}$$

$$\boxed{\tilde{\epsilon}(x_t, c) = (1 + w)\epsilon_\theta(x_t, c) - w\epsilon_\theta(x_t)} \tag{14}$$

Classifier-Free Guidance: Training Strategy

Key idea: Train ONE U-Net for both conditional and unconditional generation

Training algorithm:

Algorithm 2 Joint Training for CFG

Require: training data (x, c), unconditional dropout probability p_{uncond}

```
repeat
    for each batch (x, c) do
        With probability p_uncond: set c ← ∅ (null token)
        Sample timestep: t ~ Uniform(1, T)
        Sample noise: ϵ ~ N(0, I)
        Corrupt: x_t = sqrt(ᾱ_t) x + sqrt(1 − ᾱ_t) ϵ
        Compute loss: L = ||ϵ_θ(x_t, t, c) − ϵ||_2^2
        Update: θ ← θ − η∇_θ L
    end for
until converged
```

- With probability p_{uncond}: set $c \leftarrow \varnothing$ (null token)
- Sample timestep: $t \sim \text{Uniform}(1, T)$
- Sample noise: $\epsilon \sim \mathcal{N}(0, I)$
- Corrupt: $x_t = \sqrt{\bar{\alpha}_t}x + \sqrt{1-\bar{\alpha}_t}\epsilon$
- Compute loss: $\mathcal{L} = \|\epsilon_\theta(x_t, t, c) - \epsilon\|_2^2$
- Update: $\theta \leftarrow \theta - \eta\nabla_\theta\mathcal{L}$

Classifier-Free Guidance: Inference Algorithm

Algorithm 3 Classifier-Free Guidance Sampling

Require: condition c, guidance strength w, timesteps $\{t_1, \ldots, t_T\}$

1: **Initialize:** $x_T \sim \mathcal{N}(0, I)$
2: **for** $t = T$ to 1 **do**
3: $\quad \epsilon_{\text{cond}} \leftarrow \epsilon_\theta(x_t, t, c)$ // Conditional prediction
4: $\quad \epsilon_{\text{uncond}} \leftarrow \epsilon_\theta(x_t, t, \varnothing)$ // Unconditional prediction
5: $\quad \tilde{\epsilon} \leftarrow (1 + w)\epsilon_{\text{cond}} - w\epsilon_{\text{uncond}}$ // Linear combination
6: $\quad x_{t-1} \sim p_\theta(x_{t-1} \mid x_t, c, \tilde{\epsilon})$ // Update step
7: **end for**
8: **return** $x_0 = x_1$

Computational requirements per timestep:

- ✓ 2 forward passes: same U-Net (conditional + unconditional)
- × 0 classifier models
- × 0 gradient computations
- ✓ 1 linear combination (trivial)

Comparison: CG vs CFG + Practical Guidance Values

Aspect	**Classifier Guidance (CG)**	**Classifier-Free (CFG)**
Inference cost	2 forwards + 1 backward	2 forwards
Models needed	Diffusion + Classifier	Diffusion only
Training complexity	High (classifier on noisy)	Low (random dropout)
Peak metrics (FID/IS)	**Higher**	Competitive
Industry adoption	Limited	**Standard** (Stable Diffusion, SDXL)
Artifacts at high w	Adversarial edges	Color saturation

Inverse Problems with Diffusion

The Goal: Recover clean image x from degraded measurement $y = f(x) + n$

Challenge: Standard DDPM samples from prior $p(x)$ (random images)
Requirement: Sample from posterior $p(x|y)$ (conditional on measurement)

DDPM Limitation:

- × Ignores the observation y
- × Generates images unrelated to input
- ✓ Solution: Add **data fidelity constraint**

Combining Scores with Bayes' Rule

We want to sample from posterior $p(x_t|y)$ instead of prior $p(x_t)$.

Bayes' theorem applied to score:

$$\nabla_{x_t} \log p(x_t|y) = \underbrace{\nabla_{x_t} \log p(x_t)}_{\text{Prior}} + \underbrace{\nabla_{x_t} \log p(y|x_t)}_{\text{Likelihood}} \tag{15}$$

Two terms:

- **Prior term:** $\nabla_{x_t} \log p(x_t)$ — Given by U-Net $\epsilon_\theta(x_t)$
  - Says: "This direction is realistic"
- **Likelihood term:** $\nabla_{x_t} \log p(y|x_t)$ — Intractable!
  - Says: "This direction respects measurement y"
  - Problem: x_t is noisy; need to integrate over all possible clean x_0

Diffusion Posterior Sampling (DPS): The Approximation

Key Idea: Use the model's current denoising estimate to approximate likelihood

At each step t, the U-Net denoising x_t implicitly predicts clean image: $\hat{x}_0(x_t)$ (Tweedie estimate)

Approximate the likelihood:

$$p(y|x_t) \approx p(y \mid \hat{x}_0(x_t)) \tag{16}$$

Computing the likelihood gradient:

1. Calculate measurement error: $\mathcal{D} = \|y - f(\hat{x}_0)\|^2$
2. Compute gradient via backpropagation:

$$\nabla_{x_t} \log p(y|x_t) \approx -\zeta \cdot \nabla_{x_t} \mathcal{D} \tag{17}$$

Requires: Backpropagation through the U-Net $\rightarrow$ x_t becomes differentiable variable

Applications & Conclusion

Key Takeaways:

1. **Zero-shot:** No retraining, define f at inference
2. **General:** Linear, non-linear, any measurement operator
3. **Robust:** Soft constraints handle noisy measurements naturally
4. **Cost:** Expensive, but generalizes where specialists fail

References I

Dhariwal, P., & Nichol, A. (2021).
"Diffusion Models Beat GANs on Image Synthesis."
arXiv preprint arXiv:2105.05233.

Ho, J., & Salimans, T. (2021).
"Classifier-Free Diffusion Guidance."
NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications.

Ho, J., Jain, A., & Abbeel, P. (2020).
"Denoising Diffusion Probabilistic Models."
Advances in Neural Information Processing Systems (NeurIPS).

Song, Y., Garg, S., Gao, J., Lindenbaum, O., Tripathi, A., & Ermon, S. (2021).
"Solving Inverse Problems Deterministically without Diffusion."
arXiv preprint arXiv:2108.08713.