Diffusion Models: Theory and Applications

Lecture 5: Sampling from Trained Diffusion Models - From Noise to Data

Shubham Chatterjee

Missouri University of Science and Technology, Department of Computer Science

June 11, 2025

Last Time: We Learned How to Train Diffusion Models

We mastered the ELBO framework and noise prediction...

But how do we actually USE these trained models? 🚀

- ✓ **ELBO decomposition:** Three interpretable forces
- ✓ **Noise prediction:** Simple MSE training objective
- ✓ **Training algorithm:** From complex theory to practical implementation
- 🚀 **Today:** Turn trained models into sample generators!

Today's Mission: The Art of Generation

How do we go from pure noise to realistic data?

The Challenge:

- We have a trained noise predictor $\epsilon_\theta(\mathbf{x}_t, t)$
- Need to reverse the destruction process
- Want high quality AND reasonable speed

Two Main Approaches:

- **DDPM:** Stochastic, high quality, slow
- **DDIM:** Deterministic, fast, still high quality

Sampling = Where theory meets practice!

The Fundamental Sampling Challenge

We've trained our network to predict noise...

What we have

- Trained network: $\epsilon_\theta(\mathbf{x}_t, t)$ predicts noise
- Noise schedule: $\{\beta_t\}$ and derived $\{\alpha_t, \bar{\alpha}_t\}$
- Mathematical framework: Reverse process theory

What we need to figure out

- How to start: Where does $\mathbf{x}_T$ come from?
- How to step: What's the update rule for $\mathbf{x}_t \to \mathbf{x}_{t-1}$?
- How many steps: Can we go faster than 1000 timesteps?
- How much randomness: Should sampling be deterministic or stochastic?

Let's build the bridge from trained models to generated samples!

Reverse Process Recap: Our Mathematical Foundation

Remember how we parameterized the reverse process?

The learned reverse transition

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \tilde{\sigma}_t^2\mathbf{I})$$

Mean parameterization (from noise prediction)

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

Fixed variance

$$\tilde{\sigma}_t^2 = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

Key insight: We learn the mean, variance comes from the noise schedule! 🔑

DDPM Sampling: Following the Stochastic Path

The original sampling approach: embrace the randomness!

```
Input: Trained model $\epsilon_\theta$, timesteps $T$
Sample $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$                ▷ Start from pure noise
for $t = T, T-1, \ldots, 1$ do
    Predict noise: $\hat{\boldsymbol{\epsilon}} = \epsilon_\theta(\mathbf{x}_t, t)$
    Compute mean: $\boldsymbol{\mu} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\hat{\boldsymbol{\epsilon}}\right)$
    if $t > 1$ then
        Sample noise: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
        $\mathbf{x}_{t-1} = \boldsymbol{\mu} + \sqrt{\tilde{\sigma}_t^2} \cdot \mathbf{z}$
    else
        $\mathbf{x}_0 = \boldsymbol{\mu}$                ▷ No noise in final step
    end if
end for
Return: $\mathbf{x}_0$
```

1000 steps later... We have our generated sample!

Understanding Each Step of DDPM Sampling

1. Initialization: Start from chaos

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ - Pure Gaussian noise

2. Noise prediction: What corruption was added?

$\hat{\epsilon} = \epsilon_\theta(\mathbf{x}_t, t)$ - Use our trained network

3. Mean computation: Where should we go?

Remove predicted noise to get expected previous state

4. Stochastic sampling: Add controlled randomness

Don't just use the mean - add calibrated noise for diversity

5. Final step: Clean finish

At $t = 1$, typically don't add noise (deterministic final step)

The DDPM Trade-offs: Quality vs. Speed

✓ DDPM Strengths:

- **High quality:** Excellent samples with many steps
- **Diversity:** Stochastic sampling ensures variety
- **Theory:** Provably samples from correct distribution
- **Robustness:** Works across datasets and architectures

✖ DDPM Weaknesses:

- **Slow:** Requires 1000 function evaluations
- **Expensive:** Each step needs full network forward pass
- **Fixed:** Hard to trade quality for speed
- **Sequential:** Can't easily parallelize steps

The bottleneck: 1000 steps $\times$ expensive network $=$ too slow for real applications!

Can we do better? *Enter DDIM...*

DDIM: A Revolutionary Insight

What if we made sampling deterministic?

The key realization...

DDIM: A Revolutionary Insight

What if we made sampling deterministic?

The key realization...

DDIM's brilliant insight

DDPM defines ONE way to reverse the process...
But there are MANY reverse processes with the same marginals!

DDIM: A Revolutionary Insight

What if we made sampling deterministic?

The key realization...

DDIM's brilliant insight

DDPM defines ONE way to reverse the process...
But there are MANY reverse processes with the same marginals!

What this means

- Same training, same network, same forward process
- But different (faster!) sampling procedure
- Maintains quality while dramatically improving speed
- **Can skip steps without retraining!**

This changes everything!

DDIM's Mathematical Foundation

The key mathematical insight behind DDIM...

DDPM approach

Uses the Markovian reverse: $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$

DDIM approach

Uses the more general: $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$
Condition on BOTH current state AND original data!

The constraint

We require the same marginals: $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$

Result: A family of reverse processes parameterized by σ_t! ★

The Mathematical Surprise: σ_t Emerges!

What happens when we solve the DDIM constraint problem?

The constraint we impose

Find $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ such that:
$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$ (same as DDPM)

The mathematical discovery

When we solve this constraint (using Bayes rule + Gaussian algebra), we discover:

The solution is not unique!

There's a **family of valid solutions** parameterized by the variance σ_t^2

The Mathematical Surprise: σ_t Emerges!

The mathematical discovery

When we solve this constraint (using Bayes rule + Gaussian algebra), we discover:

The solution is not unique!

There's a **family of valid solutions** parameterized by the variance σ_t^2

DDIM's key insight

$$q_\sigma(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}(\mathbf{x}_t, \mathbf{x}_0), \sigma_t^2\mathbf{I})$$

σ_t^2 can be *any* value in range $[0, \beta_t]$ - it's a **free parameter**!

Surprise: This gives us unexpected control over randomness!

When $\sigma_t = 0$: What Happens? ?

The deterministic case - no randomness!

The big question

If we remove all randomness from the reverse process, how do we actually update from $\mathbf{x}_t$ to $\mathbf{x}_{t-1}$?

DDIM's key insight ◎

Use the noise schedule structure!

We know: $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$

If we can estimate $\mathbf{x}_0$ and ϵ, we can construct $\mathbf{x}_{t-1}$ directly!

Next: Let's see how this construction works step by step... →

The Noise Schedule: Our Mathematical Foundation

Both DDPM and DDIM rely on this fundamental relationship:

The universal noise schedule formula

At *any* timestep s, a noisy image has exactly this structure:

$$\mathbf{x}_s = \sqrt{\bar{\alpha}_s}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_s}\boldsymbol{\epsilon}$$

What this means

- $\mathbf{x}_0$: The original clean image
- $\boldsymbol{\epsilon}$: The noise vector
- $\sqrt{\bar{\alpha}_s}$: How much "signal" remains at timestep s
- $\sqrt{1-\bar{\alpha}_s}$: How much "noise" is present at timestep s

DDIM's insight: If we know $\mathbf{x}_0$ and $\boldsymbol{\epsilon}$, we can construct *any* timestep!

Step 1: Reverse Engineer the Components

Given $\mathbf{x}_t$, figure out what $\mathbf{x}_0$ and ϵ should be

Estimate the noise

Use our trained neural network:

$$\hat{\boldsymbol{\epsilon}} = \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$$

Solve for the clean image

Rearrange the noise schedule formula to find $\mathbf{x}_0$:

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\hat{\boldsymbol{\epsilon}}$$

$$\Downarrow$$

$$\hat{\mathbf{x}}_0 = \frac{\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t}\hat{\boldsymbol{\epsilon}}}{\sqrt{\bar{\alpha}_t}}$$

Now we have both pieces: $\hat{\mathbf{x}}_0$ and $\hat{\boldsymbol{\epsilon}}$ ✓

Step 2: Construct the Target Timestep

Use the same noise schedule formula to build $\mathbf{x}_{t-1}$

Apply the noise schedule at timestep $t-1$

We want: $\mathbf{x}_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}}\epsilon$

Substitute our estimates

Replace $\mathbf{x}_0$ and ϵ with our predictions:

$$\mathbf{x}_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\hat{\mathbf{x}}_0 + \sqrt{1-\bar{\alpha}_{t-1}}\hat{\epsilon}$$

Why this works

- We're using the **same underlying clean image** $\hat{\mathbf{x}}_0$
- We're using the **same noise vector** $\hat{\epsilon}$
- We're just **adjusting the noise level** for timestep $t-1$

The Complete DDIM Formula: Putting It Together

Substitute Step 1 into Step 2:

The DDIM update rule (= 0)

$$\mathbf{x}_{t-1} = \sqrt{\bar{\alpha}_{t-1}} \underbrace{\frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(\mathbf{x}_t, t)}{\sqrt{\bar{\alpha}_t}}}_{\hat{\mathbf{x}}_0 \text{ (estimated clean image)}} + \sqrt{1-\bar{\alpha}_{t-1}}\epsilon_\theta(\mathbf{x}_t, t)$$

The logic is crystal clear

1. **Estimate:** What are the clean image and noise?
2. **Construct:** Use these estimates to build the next timestep
3. **Consistency:** Same noise, different noise level

Key insight: We're not "removing then adding" noise - we're **reconstructing** the trajectory!

DDIM vs DDPM: Fundamentally Different Approaches

Two completely different philosophies:

DDPM Philosophy

"Remove Noise Gradually"

- Take tiny steps
- Remove a bit of noise each time
- Hope randomness averages out
- Many careful iterations

Like chiseling a sculpture

DDIM Philosophy

"Reconstruct the Path"

- Estimate the clean image
- Use noise schedule structure
- Jump directly to target
- Fewer confident steps

Like using a blueprint

Result: DDIM achieves similar quality with 10-50x fewer steps! ★

DDIM Sampling Algorithm: Speed and Elegance 🚀

```
**Input:** Trained model $\epsilon_\theta$, timestep subset $\{\tau_1, \tau_2, \ldots, \tau_S\}$
Sample $\mathbf{x}_{\tau_S} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $i = S, S-1, \ldots, 1$ do
    $t = \tau_i$, $s = \tau_{i-1}$ (where $\tau_0 = 0$)
    Predict noise: $\hat{\epsilon} = \epsilon_\theta(\mathbf{x}_t, t)$
    Predict $\mathbf{x}_0$: $\hat{\mathbf{x}}_0 = \frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\hat{\epsilon}}{\sqrt{\bar{\alpha}_t}}$
    Update: $\mathbf{x}_s = \sqrt{\bar{\alpha}_s}\hat{\mathbf{x}}_0 + \sqrt{1-\bar{\alpha}_s}\hat{\epsilon}$
end for
Return: $\mathbf{x}_0$
```

Key differences from DDPM: 🔑

- **Timestep subset:** Use only $S \ll T$ steps (e.g., 50 instead of 1000)
- **No random sampling:** Completely deterministic
- **Direct prediction:** Explicitly predict $\mathbf{x}_0$ then re-noise

DDPM vs. DDIM: The Ultimate Showdown 🏆

When should you use which method?

✓ **Use DDPM when:**

- Sample diversity is crucial
- Computational time isn't a constraint
- Need highest possible quality
- Theoretical guarantees matter
- Working with research/small-scale

Typical settings:

- 1000 steps
- Stochastic sampling
- Research applications

🚀 **Use DDIM when:**

- Speed is important
- Need deterministic generation
- High-resolution images
- Real-time applications
- Production deployment

Typical settings:

- 20-50 steps
- $\eta = 0$ or small
- Production applications

Modern default: DDIM with 50 steps for most applications! ★

Preview: What's Coming Next

We can now generate unconditional samples...
But what if we want CONTROL?

Conditional generation challenges

- Generate specific classes: "Give me a cat image"
- Text-to-image: "A sunset over mountains"
- Style control: "In the style of Van Gogh"
- Spatial control: "Put the object here"

Next lecture preview: Guided Sampling

- **Classifier guidance:** Use external classifiers to steer generation
- **Classifier-free guidance:** The breakthrough behind Stable Diffusion
- **Text conditioning:** How CLIP embeddings guide the process
- **Advanced control:** ControlNet and spatial conditioning

From random generation to controllable creation!

Summary: Sampling Mastery Achieved

What we've learned today

- **DDPM sampling:** Stochastic reverse process with theoretical guarantees
- **DDIM sampling:** Deterministic alternative enabling dramatic acceleration
- **Trade-offs:** Quality, speed, and diversity relationships
- **Implementation:** Practical considerations for robust sampling

Key practical insights

- Modern default: DDIM with 20-50 steps for most applications
- Use η parameter to control stochasticity vs. speed trade-off
- Deterministic sampling enables reproducibility and faster iteration
- Higher-order methods and optimizations provide further improvements

We can now turn trained diffusion models into practical sample generators!

Coming Up Next: Controlled Generation

Next Session Preview: Conditional Generation & Guidance

We'll explore how to control what diffusion models generate:

- How do we condition generation on classes, text, or images?
- What is classifier guidance and why does it work?
- How does classifier-free guidance eliminate the need for separate classifiers?
- What makes text-to-image generation possible?

From unconditional noise-to-data generation to controllable, guided creation!

Ready to add steering wheels to our diffusion models?