UPPSALA UNIVERSITET

Machine learning – Block 7c

Måns Magnusson
Department of Statistics, Uppsala University

Autumn 2025

UPPSALA UNIVERSITET

Diffusion Models

- A new class of deep probabilistic generative models

UPPSALA
UNIVERSITET

Diffusion Models

- A new class of deep probabilistic generative models
- State-of-the-art for image generation

UPPSALA UNIVERSITET

Diffusion Models

- A new class of deep probabilistic generative models
- State-of-the-art for image generation
- Reading: Bishop & Bishop, Chapter 20.

UPPSALA UNIVERSITET

Diffusion Models

- A new class of deep probabilistic generative models
- State-of-the-art for image generation
- Reading: Bishop & Bishop, Chapter 20.
- Closely related to variational autoencoders

Generative Models Recap

- (Deep) latent variable models:
  - Variational Autoencoders (VAEs)
  - Diffusion Models
- All transform a simple (latent) distribution into complex data

Generative Models Recap

- (Deep) latent variable models:
  - Variational Autoencoders (VAEs)
  - Diffusion Models
- All transform a simple (latent) distribution into complex data
- Main difference: how this transformation is learned

Why Diffusion Models?

- Stable training
- Excellent sample quality
- Main drawback: slow sampling

Demo: DALL·E 3: https://openai.com/dall-e-3

High-Level Idea

- Two processes:
  1. **Forward process:** gradually add noise to data
  2. **Reverse process:** learn to remove noise

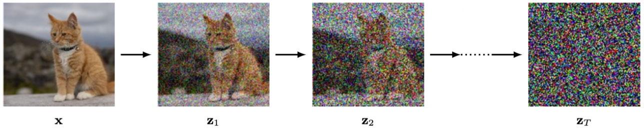


Figure: Figure 20.1 from Bishop & Bishop (2024).

UPPSALA UNIVERSITET

High-Level Idea

- Two processes:
  1. **Forward process:** gradually add noise to data
  2. **Reverse process:** learn to remove noise

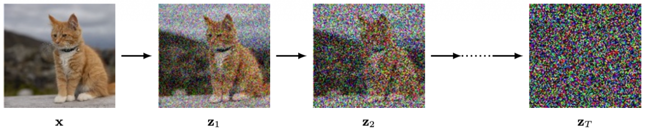


Figure: Figure 20.1 from Bishop & Bishop (2024).

- Generation starts from pure noise
- Data is generated by iteratively denoising

UPPSALA UNIVERSITET

Diffusion Models and VAEs

- Diffusion models can be viewed as hierarchical VAEs
- Key differences:
  - Encoder is fixed (noise process)

UPPSALA UNIVERSITET

Diffusion Models and VAEs

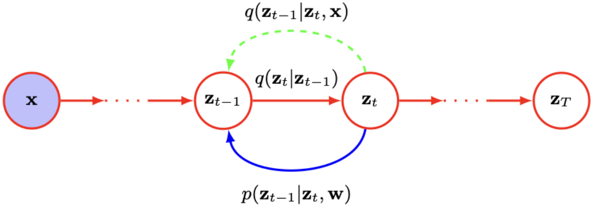
- Diffusion models can be viewed as hierarchical VAEs
- Key differences:
  - Encoder is fixed (noise process)
  - Decoder is learned (denoising network)

Diffusion Models and VAEs

- Diffusion models can be viewed as hierarchical VAEs
- Key differences:
  - Encoder is fixed (noise process)
  - Decoder is learned (denoising network)
  - Many latent variables instead of a few

UPPSALA UNIVERSITET

Forward Diffusion Process



$q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})$
$\mathbf{x}$
$\mathbf{z}_{t-1}$
$q(\mathbf{z}_t|\mathbf{z}_{t-1})$
$\mathbf{z}_t$
$\mathbf{z}_T$
$p(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{w})$

Figure: Figure 20.2 from Bishop & Bishop (2024).

- Start with data point x
- Add small Gaussian noise repeatedly
- After many steps: result is Gaussian noise

UPPSALA UNIVERSITET

Forward Process: Single Step

For $t = 1, \ldots, T$:

$$z_t = \sqrt{1-\beta_t} z_{t-1} + \sqrt{\beta_t}\, \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, I)$$

- β_t controls noise level
- Noise increases with t
- Defines a Markov chain

UPPSALA UNIVERSITET

Probabilistic Encoder

- The forward process defines:

$$q(z_t \mid z_{t-1}) = \mathcal{N}\left(\sqrt{1-\beta_t}\, z_{t-1}, \beta_t I\right)$$

- This plays the role of an **encoder**
- Unlike VAEs: encoder is fixed

UPPSALA UNIVERSITET

Closed-Form Noising

Key result:

$$z_t = \sqrt{\alpha_t} x + \sqrt{1 - \alpha_t}\, \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I)$$

- $\alpha_t = \prod_{\tau=1}^{t} (1 - \beta_\tau)$
- Allows direct sampling of z_t from x
- Very important for training

UPPSALA UNIVERSITET

End of Forward Process

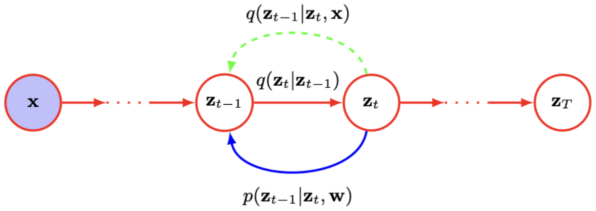
- As $t \to T$:

$$z_T \sim \mathcal{N}(0, I)$$

- All information about x is lost
- This distribution is easy to sample from

UPPSALA UNIVERSITET

Learning the Reverse Process



$q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})$
$\mathbf{x}$
$\mathbf{z}_{t-1}$
$q(\mathbf{z}_t|\mathbf{z}_{t-1})$
$\mathbf{z}_t$
$\mathbf{z}_T$
$p(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{w})$

Figure: Figure 20.2 from Bishop & Bishop (2024).

- Goal: reverse the noise process

$$p(z_{t-1} \mid z_t)$$

- Exact reverse distribution is intractable
- We learn an approximation using a neural network

UPPSALA UNIVERSITET

Probabilistic Decoder

We model:

$$p(z_{t-1} \mid z_t, w) = \mathcal{N}(\mu(z_t, t), \beta_t I)$$

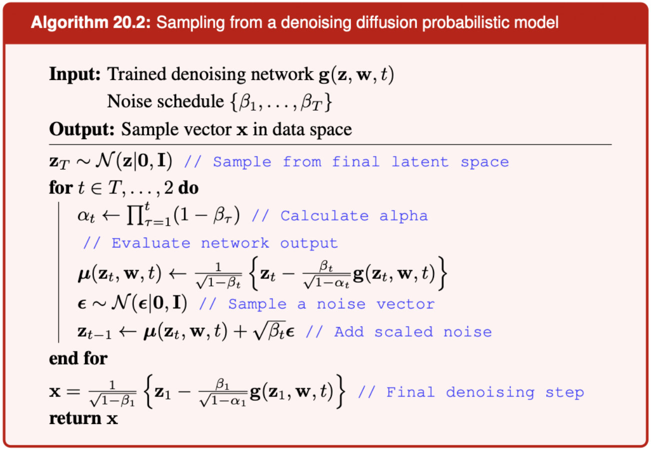
- Mean predicted by neural network
- Variance often fixed
- Decoder shared across all steps

UPPSALA UNIVERSITET

Sampling from the Model

1. Sample $z_T \sim \mathcal{N}(0, I)$
2. For $t = T, \ldots, 1$:
    - Sample $z_{t-1} \sim p(z_{t-1} \mid z_t)$
3. Final sample corresponds to data point x

Sampling from the Model



Algorithm 20.2: Sampling from a denoising diffusion probabilistic model

```
Input: Trained denoising network $\mathbf{g}(\mathbf{z}, \mathbf{w}, t)$
       Noise schedule $\{\beta_1, \dots, \beta_T\}$
Output: Sample vector $\mathbf{x}$ in data space
$\mathbf{z}_T \sim \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I})$ // Sample from final latent space
for $t \in T, \dots, 2$ do
    $\alpha_t \leftarrow \prod_{\tau=1}^{t}(1 - \beta_\tau)$ // Calculate alpha
    // Evaluate network output
    $\boldsymbol{\mu}(\mathbf{z}_t, \mathbf{w}, t) \leftarrow \frac{1}{\sqrt{1-\beta_t}}\left\{\mathbf{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\mathbf{g}(\mathbf{z}_t, \mathbf{w}, t)\right\}$
    $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0}, \mathbf{I})$ // Sample a noise vector
    $\mathbf{z}_{t-1} \leftarrow \boldsymbol{\mu}(\mathbf{z}_t, \mathbf{w}, t) + \sqrt{\beta_t}\boldsymbol{\epsilon}$ // Add scaled noise
end for
$\mathbf{x} = \frac{1}{\sqrt{1-\beta_1}}\left\{\mathbf{z}_1 - \frac{\beta_1}{\sqrt{1-\alpha_1}}\mathbf{g}(\mathbf{z}_1, \mathbf{w}, t)\right\}$ // Final denoising step
return $\mathbf{x}$
```

Figure: Algorithm 20.2 from Bishop & Bishop (2024).

UPPSALA UNIVERSITET

Training Objective

- Exact likelihood is intractable
- Maximize an Evidence Lower Bound (ELBO)
- Very similar to VAE training

UPPSALA
UNIVERSITET

ELBO Interpretation

- Reconstruction term
- Consistency terms between forward and reverse processes
- Encoder distribution is fixed
- Only decoder parameters are learned

Predicting Noise Instead of Data

Key idea:

- Instead of predicting z_{t-1}
- Predict the noise ε added to x: $g(z_t, t)$

UPPSALA UNIVERSITET

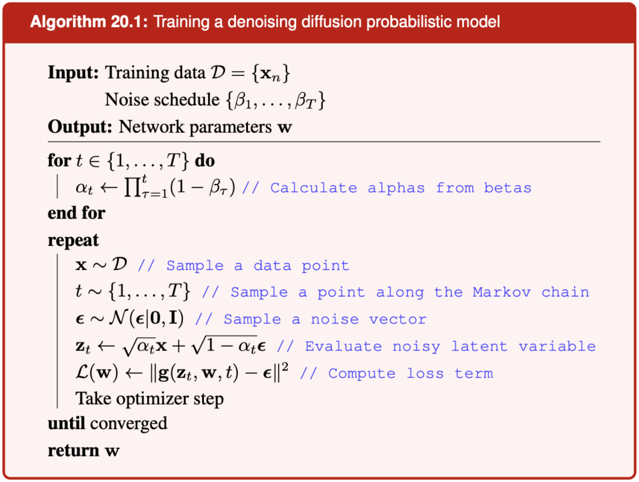
Final Training Loss

The loss simplifies to:

$$\mathbb{E}_{x,t,\varepsilon}\left[\|\varepsilon - g(z_t, t)\|^2\right]$$

- Simple squared error loss
- Very stable optimization
- Central result of diffusion models

Training a Denoising diffusion model



Algorithm 20.1: Training a denoising diffusion probabilistic model

Input: Training data $\mathcal{D} = \{\mathbf{x}_n\}$
Noise schedule $\{\beta_1, \ldots, \beta_T\}$
Output: Network parameters $\mathbf{w}$

```
for $t \in \{1, \ldots, T\}$ do
    $\alpha_t \leftarrow \prod_{\tau=1}^{t}(1 - \beta_\tau)$ // Calculate alphas from betas
end for
repeat
    $\mathbf{x} \sim \mathcal{D}$ // Sample a data point
    $t \sim \{1, \ldots, T\}$ // Sample a point along the Markov chain
    $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0}, \mathbf{I})$ // Sample a noise vector
    $\mathbf{z}_t \leftarrow \sqrt{\alpha_t}\mathbf{x} + \sqrt{1 - \alpha_t}\boldsymbol{\epsilon}$ // Evaluate noisy latent variable
    $\mathcal{L}(\mathbf{w}) \leftarrow \|\mathbf{g}(\mathbf{z}_t, \mathbf{w}, t) - \boldsymbol{\epsilon}\|^2$ // Compute loss term
    Take optimizer step
until converged
return $\mathbf{w}$
```

Figure: Algorithm 20.1 from Bishop & Bishop (2024).

Diffusion Models vs VAEs

	VAE	Diffusion
Encoder	Learned	Fixed
Latent dim.	Low	High
Decoder	One step	Many steps
Training	ELBO	ELBO
Sampling	Fast	Slow

Conditional Diffusion Models

- So far: unconditional generation
  - Sample images from $p(x)$
- Often we want conditional generation
  - Generate images given text
  - Generate images given class labels
- Goal: model $p(x \mid c)$, where c is conditioning information

UPPSALA UNIVERSITET

Conditioning on Text

- Let c be a text description
- Text is encoded using a language model

$$c \rightarrow \text{embedding } e(c)$$

- Diffusion model is conditioned on this embedding
- Reverse process becomes:

$$p(z_{t-1} \mid z_t, c)$$

UPPSALA
UNIVERSITET

How Conditioning Is Used

- Conditioning information is provided to the neural network
- Noise prediction network becomes:

$$g(z_t, t, c)$$

- Intuition:
  - Noise removal is guided by the text
  - Different texts lead to different denoising trajectories
- Used in systems such as:
  - Text-to-image generation
  - Image editing and inpainting

UPPSALA UNIVERSITET

How Conditioning Is Used

- Used in systems such as:
  - Text-to-image generation
  - Image editing and inpainting

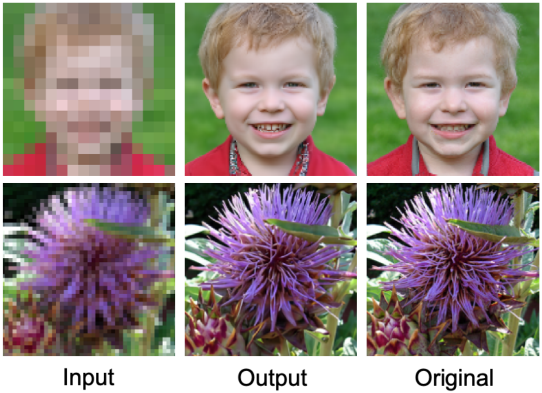


Figure: Figure 20.8 from Bishop & Bishop (2024).

UPPSALA UNIVERSITET

Diffusion Ideas Beyond Images

- Diffusion models popularized a key idea:
  - Learn complex objects by iterative denoising

Diffusion Ideas Beyond Images

- Diffusion models popularized a key idea:
  - Learn complex objects by iterative denoising
- AlphaFold 2 applies a related principle to protein structures:
  - Protein geometry is refined step-by-step
  - Each step reduces uncertainty and inconsistency
- Important distinction:
  - AlphaFold 2 is not a Denoising Diffusion Probabilistic Model
  - But it uses a diffusion-*like* denoising process

UPPSALA UNIVERSITET

Diffusion Ideas Beyond Images

- Diffusion models popularized a key idea:
  - Learn complex objects by iterative denoising
- AlphaFold 2 applies a related principle to protein structures:
  - Protein geometry is refined step-by-step
  - Each step reduces uncertainty and inconsistency
- Important distinction:
  - AlphaFold 2 is not a Denoising Diffusion Probabilistic Model
  - But it uses a diffusion-*like* denoising process
- New generative biology models now use explicit diffusion models
  - Protein backbone generation (Watson et al, 2023)
  - Molecular design (Hoogeboom et al, 2022)

Key Takeaways

- Diffusion models are deep probabilistic models
- Closely related to VAEs
- Built around denoising Gaussian noise
- Foundation for modern image generation systems conditional on textual input