Lecture 01 - Deep Generative Modeling

Baojian Zhou

Topics on Diffusion Models

School of Data Science, Fudan University

The book

https://arxiv.org/abs/2510.21890

The Principles of Diffusion Models

From Origins to Advances

Chieh-Hsin Lai
Sony AI

Yang Song
OpenAI

Dongjun Kim
Stanford University

Yuki Mitsufuji
Sony Corporation, Sony AI

Stefano Ermon
Stanford University

1. Chieh-Hsin (Jesse) Lai
   - Research Scientist at Sony AI's Music Foundation Model Team
   - Ph.D. in Mathematics from the University of Minnesota, Twin Cities
   - Latest Publications

1. Chieh-Hsin (Jesse) Lai
   - Research Scientist at Sony AI's Music Foundation Model Team
   - Ph.D. in Mathematics from the University of Minnesota, Twin Cities
   - Latest Publications
2. Song Yang
   - Research Principal at Meta Superintelligence Labs
   - Ph.D. in CS from Stanford University, advised by Stefano Ermon
   - Latest Publications

1. Chieh-Hsin (Jesse) Lai
   - Research Scientist at Sony AI's Music Foundation Model Team
   - Ph.D. in Mathematics from the University of Minnesota, Twin Cities
   - Latest Publications
2. Song Yang
   - Research Principal at Meta Superintelligence Labs
   - Ph.D. in CS from Stanford University, advised by Stefano Ermon
   - Latest Publications
3. Dongjun Kim
   - Postdoctoral scholar in Prof. Stefano Ermon's group at Stanford
   - Ph.D at KAIST
   - Latest Publications

1. Chieh-Hsin (Jesse) Lai
   - Research Scientist at Sony AI's Music Foundation Model Team
   - Ph.D. in Mathematics from the University of Minnesota, Twin Cities
   - Latest Publications
2. Song Yang
   - Research Principal at Meta Superintelligence Labs
   - Ph.D. in CS from Stanford University, advised by Stefano Ermon
   - Latest Publications
3. Dongjun Kim
   - Postdoctoral scholar in Prof. Stefano Ermon's group at Stanford
   - Ph.D at KAIST
   - Latest Publications
4. Yuki Mitsufuji
   - Distinguished Engineer, Sony Group Corporation
   - PhD in Information Science & Technology, the University of Tokyo
   - Latest Publications

1. Chieh-Hsin (Jesse) Lai
   - Research Scientist at Sony AI's Music Foundation Model Team
   - Ph.D. in Mathematics from the University of Minnesota, Twin Cities
   - Latest Publications
2. Song Yang
   - Research Principal at Meta Superintelligence Labs
   - Ph.D. in CS from Stanford University, advised by Stefano Ermon
   - Latest Publications
3. Dongjun Kim
   - Postdoctoral scholar in Prof. Stefano Ermon's group at Stanford
   - Ph.D at KAIST
   - Latest Publications
4. Yuki Mitsufuji
   - Distinguished Engineer, Sony Group Corporation
   - PhD in Information Science & Technology, the University of Tokyo
   - Latest Publications
5. Stefano Ermon
   - Associate Professor, Department of CS, Stanford University
   - Computer Science, Cornell University
   - Latest Publications

Main content

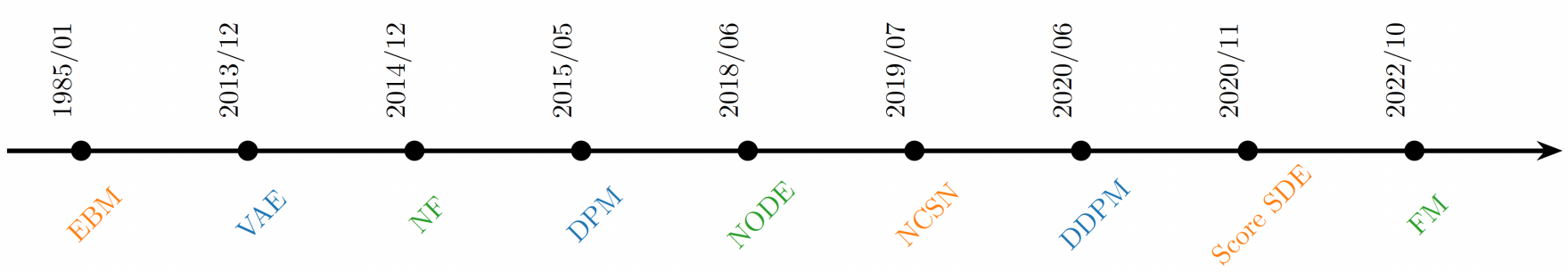
- The principles that have shaped the development of diffusion models
- Diffusion modeling starts by defining a forward process that gradually corrupts data into noise, linking the data distribution to a simple prior through a continuum of intermediate distributions.
- The main goal is to learn a reverse process that transforms noise back into data while recovering the same intermediates.

Main content

- The principles that have shaped the development of diffusion models
- Diffusion modeling starts by defining a forward process that gradually corrupts data into noise, linking the data distribution to a simple prior through a continuum of intermediate distributions.
- The main goal is to learn a reverse process that transforms noise back into data while recovering the same intermediates.
- Three complementary ways to formalize the above two processes:
  - Variational view (Variational autoencoders)
  - Score-based view (EBMs)
  - Flow-based view (Normalizing flows)

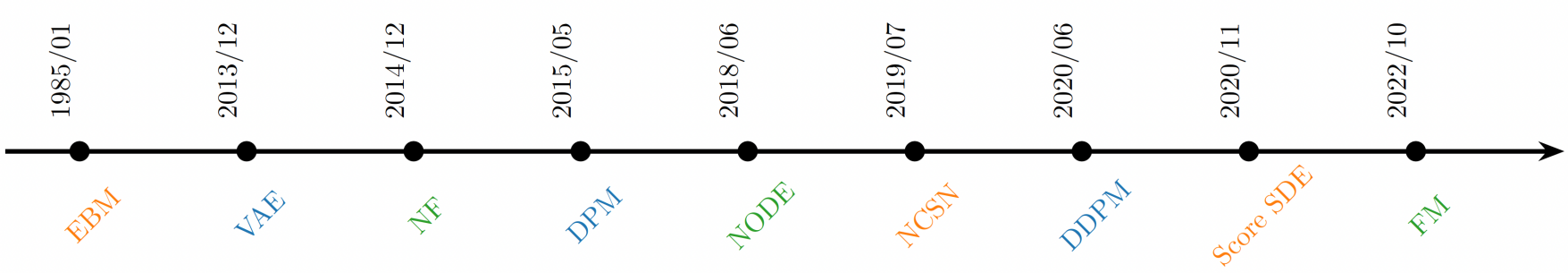
  They share a common backbone: a learned time-dependent velocity field whose flow transports a simple prior to the data.
- Sampling amounts to solving a differential equation that evolves noise into data along a continuous generative trajectory.

Timeline of diffusion model perspectives



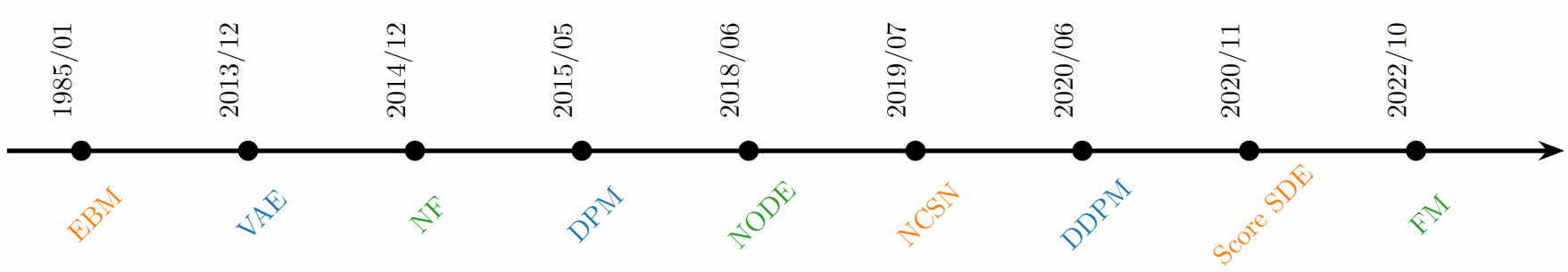
1. Score-based view
   - Energy-Based Model (EBM)[Ackley et al., 1985], Noise Conditional Score Network (NCSN)[Song and Ermon, 2019], Score SDE [Song et al., 2020]

Timeline of diffusion model perspectives



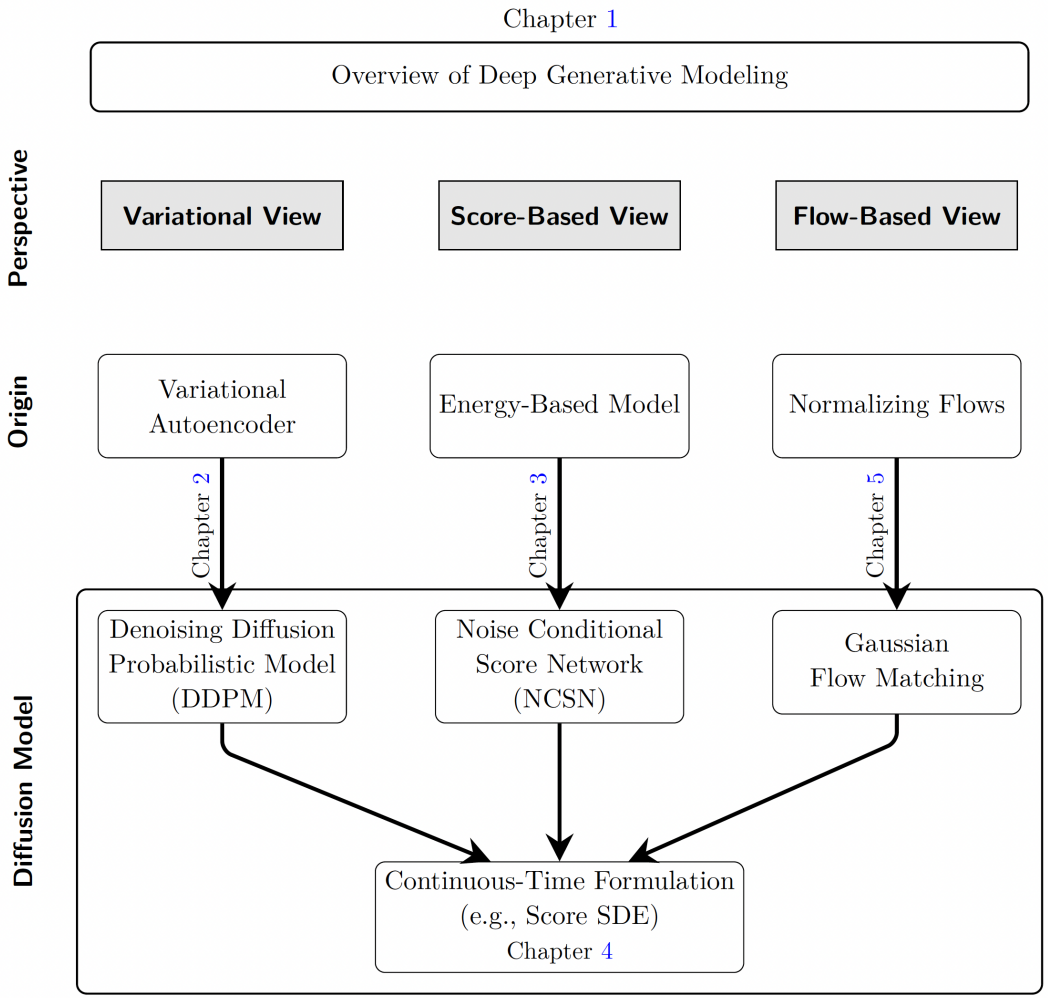
1. Score-based view
   - Energy-Based Model (EBM)[Ackley et al., 1985], Noise Conditional Score Network (NCSN)[Song and Ermon, 2019], Score SDE [Song et al., 2020]
2. Variational view
   - Variational Autoencoder (VAE) [Kingma and Welling, 2013], Diffusion Probabilistic Models (DPM) [Sohl-Dickstein et al., 2015], DDPM [Ho et al., 2020]

Timeline of diffusion model perspectives



1. Score-based view
   - Energy-Based Model (EBM)[Ackley et al., 1985], Noise Conditional Score Network (NCSN)[Song and Ermon, 2019], Score SDE [Song et al., 2020]
2. Variational view
   - Variational Autoencoder (VAE) [Kingma and Welling, 2013], Diffusion Probabilistic Models (DPM) [Sohl-Dickstein et al., 2015], DDPM [Ho et al., 2020]
3. Flow-based view
   - Normalizing Flow (NF) [Rezende and Mohamed, 2015], Neural ODE (NODE) [Chen et al., 2018], Flow-matching (FM) [Lipman et al., 2022]

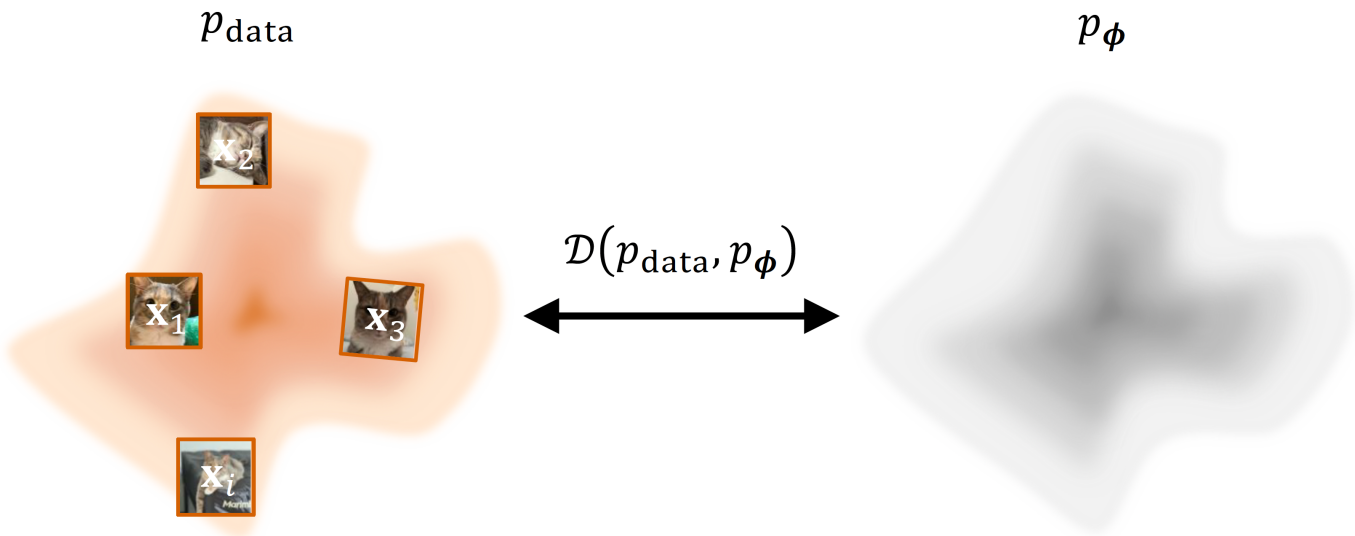
First 6 Chapters



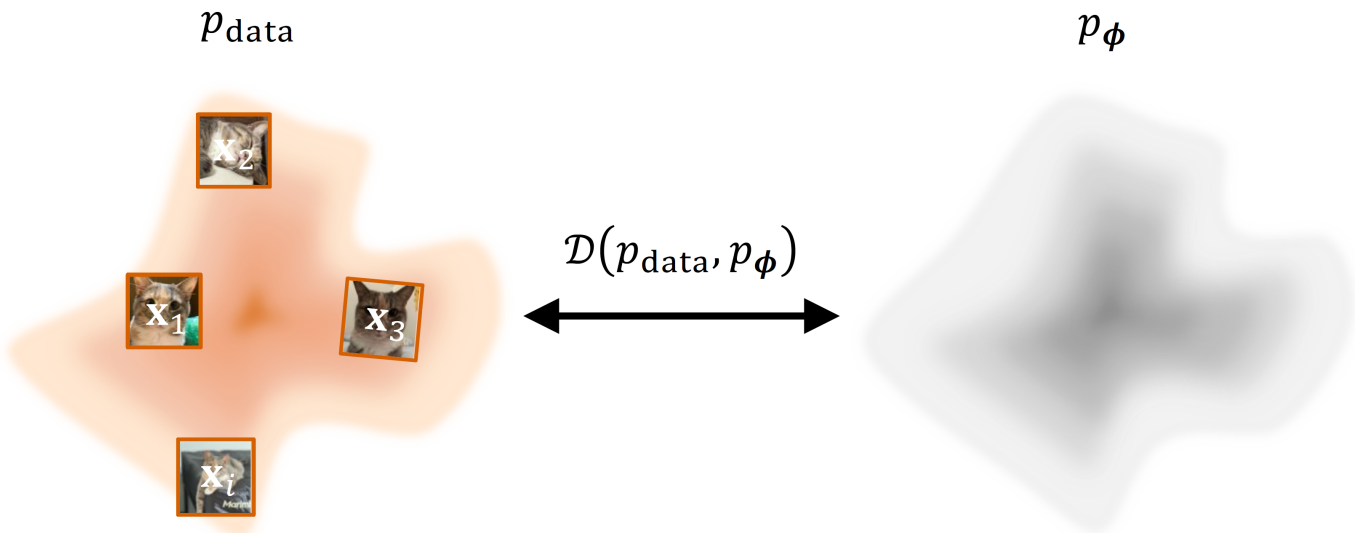
Roadmap of Our Seminar

1. Introduction to DGMs (Lecture 01, this lecture)
2. Core Perspectives on Diffusion Models
    1. Variational View (Lecture 02)
    2. Score-Based View (Lecture 03, Yu Xiang)
    3. Diffusion Models Today (Lecture 04, Zhang Hanxing)
    4. Flow-Based View (Lecture 05, Yuxiang Wang)
3. Unified and Systematic Lens on Diffusion Models (Lecture 06, Tang Jiyang)
4. Diffusion Models and Optimal Transport (Lecture 07, Jieran Lu)
5. Guidance and Controllable Generation (Lecture 08, Li Mingze, Kailin Han)
6. Sophisticated Solvers for Fast Sampling (Lecture 09, Huang binbin, Chenghao Yang)
7. Distillation-Based Methods for Fast Sampling (Lecture 10, Zhou Xinyu)
8. Learning Fast Generators from Scratch (Lecture 11)

Deep Generative Modeling



Deep Generative Modeling



- Deep generative models (DGMs) are neural networks (NNs) that learn a probability distribution over high-dimensional data (e.g., images, text, audio) so they can generate new examples that resemble the dataset.
- Training a DGM is essentially minimizing the discrepancy between the model distribution p_{ϕ} and the unknown data distribution p_{data}.

Deep Generative Modeling

The goal of learning DGMs are twofold:

- **Realistic generation**: to generate novel, realistic samples indistinguishable from real data.
- **Controllable generation**: to enable fine-grained and interpretable control over the generative process.

Deep Generative Modeling

The goal of learning DGMs are twofold:

- **Realistic generation**: to generate novel, realistic samples indistinguishable from real data.
- **Controllable generation**: to enable fine-grained and interpretable control over the generative process.

Assumptions of p_ϕ and p_{data}:

- Take a collection of examples (e.g., images, text, audio) drawn from p_{data} as input: A finite set of samples $D_{\text{tr}} := \{\boldsymbol{x}^{(i)}\}_{i=1}^{N} \overset{\text{i.i.d.}}{\sim} p_{\text{data}}$
- The true distribution p_{data} is an unknown and complex
- The learned model p_ϕ is commonly called generative model

Deep Generative Modeling

- p_ϕ: the probability distribution where ϕ are learnable NN parameters
- p_{data}: the true data distribution we want to approximate
- $\boldsymbol{x} \in \mathbb{R}^d$: a data point (an image, a sequence of text, an audio)
- $\mathcal{D}(p, q)$: measure the distance between two distributions

Deep Generative Modeling

- p_ϕ: the probability distribution where ϕ are learnable NN parameters
- p_{data}: the true data distribution we want to approximate
- $\boldsymbol{x} \in \mathbb{R}^d$: a data point (an image, a sequence of text, an audio)
- $\mathcal{D}(p, q)$: measure the distance between two distributions

Given training samples D_{tr}^T, we fit ϕ by minimizing $\mathcal{D}(p_{\text{data}}, p_\phi)$

$$\phi^* \in \arg\min_{\phi} \quad \mathcal{D}\left(p_{\text{data}}(\boldsymbol{x}), p_\phi(\boldsymbol{x})\right). \tag{1.1}$$

Hopefully, we have

$$p_{\phi^*}(\boldsymbol{x}) \approx p_{\text{data}}(\boldsymbol{x}). \tag{1.2}$$

Deep Generative Modeling

- p_ϕ: the probability distribution where ϕ are learnable NN parameters
- p_{data}: the true data distribution we want to approximate
- $\boldsymbol{x} \in \mathbb{R}^d$: a data point (an image, a sequence of text, an audio)
- $\mathcal{D}(p, q)$: measure the distance between two distributions

Given training samples D_{tr}^T, we fit ϕ by minimizing $\mathcal{D}(p_{\text{data}}, p_\phi)$

$$\phi^* \in \arg\min_{\phi} \quad \mathcal{D}\left(p_{\text{data}}(\boldsymbol{x}), p_\phi(\boldsymbol{x})\right). \tag{1.1}$$

Hopefully, we have

$$p_{\phi^*}(\boldsymbol{x}) \approx p_{\text{data}}(\boldsymbol{x}). \tag{1.2}$$

Once p_{ϕ^*} is available, we can

1. Generate $\boldsymbol{x}$: draw $\boldsymbol{x} \sim p_\phi$ via the model's sampling methods
2. Evaluate $p_\phi(\boldsymbol{x}')$: compute the probability or likelihood of any sample $\boldsymbol{x}'$

What is $\mathcal{D}$?

Choices of $\mathcal{D}$ - KL divergence

A standard choice is the (forward) Kullback-Leibler (KL) divergence

$$\begin{aligned} \mathcal{D}_{\mathrm{KL}}\left(p_{\mathrm{data}} \| p_{\boldsymbol{\phi}}\right) &:= \int p_{\mathrm{data}}(\boldsymbol{x}) \cdot \log \frac{p_{\mathrm{data}}(\boldsymbol{x})}{p_{\boldsymbol{\phi}}(\boldsymbol{x})} \mathrm{d}\boldsymbol{x} \\ &= \mathbb{E}_{\boldsymbol{x} \sim p_{\mathrm{data}}}\left[\log p_{\mathrm{data}}(\boldsymbol{x}) - \log p_{\boldsymbol{\phi}}(\boldsymbol{x})\right]. \end{aligned} \tag{1.3}$$

Choices of $\mathcal{D}$ - KL divergence

A standard choice is the (forward) Kullback-Leibler (KL) divergence

$$\begin{aligned} \mathcal{D}_{\mathrm{KL}}\left(p_{\mathrm{data}} \| p_{\boldsymbol{\phi}}\right) &:= \int p_{\mathrm{data}}(\boldsymbol{x}) \cdot \log \frac{p_{\mathrm{data}}(\boldsymbol{x})}{p_{\boldsymbol{\phi}}(\boldsymbol{x})} \mathrm{d}\boldsymbol{x} \\ &= \mathbb{E}_{\boldsymbol{x} \sim p_{\mathrm{data}}}\left[\log p_{\mathrm{data}}(\boldsymbol{x}) - \log p_{\boldsymbol{\phi}}(\boldsymbol{x})\right]. \end{aligned} \tag{1.3}$$

1. $\mathcal{D}_{\mathrm{KL}}(p\|q) \geq 0$ and $\mathcal{D}_{\mathrm{KL}}(p\|q) = 0$ iff $p = q$
2. $\mathcal{D}_{\mathrm{KL}}(p, q)$ is asymmetric: $\mathcal{D}_{\mathrm{KL}}\left(p_{\mathrm{data}}\|p_{\boldsymbol{\phi}}\right) \neq \mathcal{D}_{\mathrm{KL}}\left(p_{\boldsymbol{\phi}}\|p_{\mathrm{data}}\right)$
3. $\mathcal{D}_{\mathrm{KL}}$ is called relative entropy in information theory, where it measures how much information is lost when $p_{\boldsymbol{\phi}}$ is used to approximate p_{data}
4. Minimizing $\mathcal{D}_{\mathrm{KL}}(p_{\mathrm{data}}\|p_{\boldsymbol{\phi}})$ encourages mode covering:
   If there exists a set of positive measure A with $p_{\mathrm{data}}(A) > 0$ but $p_{\boldsymbol{\phi}}(\boldsymbol{x}) = 0$ for $\boldsymbol{x} \in A$, then the integrand contains $\log\left(p_{\mathrm{data}}(\boldsymbol{x})/0\right) = +\infty$ on A, so $\mathcal{D}_{\mathrm{KL}} = +\infty$. Thus, minimizing KL forces the model to assign probability wherever the data has support.

Choices of $\mathcal{D}$ - KL divergence

5. Minimizing KL = Minimizing cross-entropy

$$\mathcal{D}_{\mathrm{KL}}\left(p_{\mathrm{data}} \| p_{\phi}\right) = \underbrace{-\mathbb{E}_{\boldsymbol{x} \sim p_{\mathrm{data}}}\left[\log p_{\phi}(\boldsymbol{x})\right]}_{\mathcal{H}(p_{\mathrm{data}}, p_{\phi})} - \underbrace{\left(-\mathbb{E}_{\boldsymbol{x} \sim p_{\mathrm{data}}}\left[\log p_{\mathrm{data}}(\boldsymbol{x})\right]\right)}_{\mathcal{H}(p_{\mathrm{data}})}$$

$$= \mathcal{H}(p_{\mathrm{data}}, p_{\phi}) - \mathcal{H}\left(p_{\mathrm{data}}\right)$$

- $\mathcal{H}\left(p_{\mathrm{data}}\right)$ is the entropy of p_{data}
- $\mathcal{H}\left(p_{\mathrm{data}}, p_{\phi}\right)$ is the cross-entropy of p_{data} relative to p_{ϕ}

Choices of $\mathcal{D}$ - KL divergence

5. Minimizing KL = Minimizing cross-entropy

$$\mathcal{D}_{\mathrm{KL}}\left(p_{\mathrm{data}} \| p_{\phi}\right) = \underbrace{-\mathbb{E}_{\boldsymbol{x}\sim p_{\mathrm{data}}}\left[\log p_{\phi}(\boldsymbol{x})\right]}_{\mathcal{H}(p_{\mathrm{data}}, p_{\phi})} - \underbrace{\left(-\mathbb{E}_{\boldsymbol{x}\sim p_{\mathrm{data}}}\left[\log p_{\mathrm{data}}(\boldsymbol{x})\right]\right)}_{\mathcal{H}(p_{\mathrm{data}})}$$

$$= \mathcal{H}(p_{\mathrm{data}}, p_{\phi}) - \mathcal{H}\left(p_{\mathrm{data}}\right)$$

- $\mathcal{H}\left(p_{\mathrm{data}}\right)$ is the entropy of p_{data}
- $\mathcal{H}\left(p_{\mathrm{data}}, p_{\phi}\right)$ is the cross-entropy of p_{data} relative to p_{ϕ}

6. Minimizing KL $\Leftrightarrow$ Maximum likelihood estimation (MLE)

$$\min_{\phi} \mathcal{D}_{\mathrm{KL}}\left(p_{\mathrm{data}} \| p_{\phi}\right) \Longleftrightarrow \max_{\phi} \mathbb{E}_{\boldsymbol{x}\sim p_{\mathrm{data}}}\left[\log p_{\phi}(\boldsymbol{x})\right]$$

Given $D_{tr}^{N} := \left\{\boldsymbol{x}^{(i)}\right\}_{i=1}^{N} \overset{\text{i.i.d.}}{\sim} p_{\mathrm{data}}$, yielding the empirical MLE

$$\phi^{*} \in \arg\min_{\phi} \left\{ \hat{\mathcal{L}}_{\mathrm{MLE}}(\phi) := -\frac{1}{N} \sum_{i=1}^{N} \log p_{\phi}\left(\boldsymbol{x}^{(i)}\right) \right\}.$$

Choices of $\mathcal{D}$ - Fisher Divergence

Fisher Divergence: another important concept for (score-based) diffusion modeling. For two distributions p_{data} and p_{ϕ}, it is defined as

$$\mathcal{D}_{\text{F}}(p_{\text{data}} \| p_{\phi}) := \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} \left[\| \nabla_{\boldsymbol{x}} \log p_{\text{data}}(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} \log p_{\phi}(\boldsymbol{x}) \|_2^2 \right] \tag{1.4}$$

[1] A vector field is an assignment of a vector to each point in $\mathbb{R}^d$.

Choices of $\mathcal{D}$ - Fisher Divergence

Fisher Divergence: another important concept for (score-based) diffusion modeling. For two distributions p_{data} and p_ϕ, it is defined as

$$\mathcal{D}_{\text{F}}(p_{\text{data}} \| p_\phi) := \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} \left[\| \nabla_{\boldsymbol{x}} \log p_{\text{data}}(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} \log p_\phi(\boldsymbol{x}) \|_2^2 \right] \tag{1.4}$$

1. $\mathcal{D}_{\text{F}}(p_{\text{data}} \| p_\phi) \geq 0$, $\mathcal{D}_{\text{F}}(p_{\text{data}} \| p_\phi) = 0$ if and only if $p_{\text{data}} = p_\phi$.
2. The Fisher divergence $\mathcal{D}_{\text{F}}$ measures the discrepancy between the score functions $\nabla_{\boldsymbol{x}} \log p_{\text{data}}(\boldsymbol{x})$ and $\nabla_{\boldsymbol{x}} \log q_\phi(\boldsymbol{x})$, which are vector fields pointing toward regions of higher probability. [1] (score matching)
3. It is invariant to normalization constants, since scores depend only on gradients of log-densities, and it forms the basis of score matching.
4. The model p_ϕ is trained to align its score field with that of the data.

[1] A vector field is an assignment of a vector to each point in $\mathbb{R}^d$.

Choices of $\mathcal{D}$ - f divergence

Different divergences capture different geometric or statistical notions of discrepancy, which in turn affect the optimization dynamics of learning algorithms. A broad family is the f-divergences:

$$\mathcal{D}_f(p\|q) = \int q(\boldsymbol{x}) f\left(\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}\right) \mathrm{d}\boldsymbol{x}, \quad f(1) = 0, \tag{1.5}$$

where $f : \mathbb{R}_+ \to \mathbb{R}$ is a convex function. Examples of f-divergence:

1. When $f(u) = u \log u$, then $\mathcal{D}_f = \mathcal{D}_{\mathrm{KL}}$

$$f(u) = u \log u \quad \Rightarrow \quad \mathcal{D}_f = \mathcal{D}_{\mathrm{KL}}(p\|q)$$

2. Jensen-Shannon (JS) divergence (extended version see Nielsen [2025])

$$f(u) = \frac{1}{2}\left[u \log u - (u+1) \log \frac{1+u}{2}\right] \quad \Rightarrow \quad \mathcal{D}_f = \mathcal{D}_{\mathrm{JS}}(p\|q)$$

Practical choices of $\mathcal{D}$ - f divergence

3. Total Variation distance (see [Tao et al., 2024])

$$f(u) = \frac{1}{2}|u-1| \quad \Rightarrow \quad \mathcal{D}_f = \mathcal{D}_{\mathrm{TV}}(p,q) = \frac{1}{2}\int |p(\boldsymbol{x}) - q(\boldsymbol{x})| d\boldsymbol{x}$$

Properties of these f-divergences

- $\mathcal{D}_{\mathrm{JS}}$ is symmetric and its explicit form is

$$\mathcal{D}_{\mathrm{JS}}(p\|q) = \frac{1}{2}\mathcal{D}_{\mathrm{KL}}\left(p\|\tfrac{1}{2}(p+q)\right) + \frac{1}{2}\mathcal{D}_{\mathrm{KL}}\left(q\|\tfrac{1}{2}(p+q)\right).$$

- $\mathcal{D}_{\mathrm{JS}}$ is theoretically related to GAN (see Goodfellow et al. [2014])
- The explicit form of $\mathcal{D}_{\mathrm{TV}}$ is

$$\mathcal{D}_{\mathrm{TV}}(p,q) = \frac{1}{2}\int_{\mathbb{R}^D} |p(\boldsymbol{x}) - q(\boldsymbol{x})| \mathrm{d}\boldsymbol{x} = \sup_{A\subset\mathbb{R}^D} |p(A) - q(A)|.$$

Challenges in modeling distributions

To model p_{data}, we can parameterize p_{data} using NNs with parameters ϕ, creating a model we denote as p_ϕ. For p_ϕ to be a valid probability density function, it must satisfy two fundamental properties:

1. **Non-negativity**: $p_\phi(\boldsymbol{x}) \geq 0$ for all $\boldsymbol{x}$ in the domain.
2. **Normalization**: p_ϕ is normalized in domain, i.e., $\int p_\phi(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = 1$.

Challenges in modeling distributions

To model p_{data}, we can parameterize p_{data} using NNs with parameters ϕ, creating a model we denote as p_ϕ. For p_ϕ to be a valid probability density function, it must satisfy two fundamental properties:

1. **Non-negativity**: $p_\phi(\boldsymbol{x}) \geq 0$ for all $\boldsymbol{x}$ in the domain.
2. **Normalization**: p_ϕ is normalized in domain, i.e., $\int p_\phi(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = 1$.

A network can naturally produce a real scalar $E_\phi(\boldsymbol{x}) \in \mathbb{R}$ for input $\boldsymbol{x}$. To interpret this output as a valid density, it must be transformed to satisfy the above two conditions. A practical alternative is to view $E_\phi : \mathbb{R}^D \to \mathbb{R}$ as defining an unnormalized density and then enforce these properties explicitly.

Challenges in modeling distributions

Step 1: Ensuring Non-Negativity. By applying a positive function to $E_\phi(\boldsymbol{x})$, such as $|E_\phi(\boldsymbol{x})|$, $E_\phi^2(\boldsymbol{x})$, or $\exp(E_\phi(\boldsymbol{x}))$. A standard and convenient choice is the exponential function. It gives us $\tilde{p}_\phi(\boldsymbol{x})$:

$$\tilde{p}_\phi(\boldsymbol{x}) = \exp(E_\phi(\boldsymbol{x}))$$

Challenges in modeling distributions

Step 1: Ensuring Non-Negativity. By applying a positive function to $E_\phi(\boldsymbol{x})$, such as $|E_\phi(\boldsymbol{x})|$, $E_\phi^2(\boldsymbol{x})$, or $\exp(E_\phi(\boldsymbol{x}))$. A standard and convenient choice is the exponential function. It gives us $\tilde{p}_\phi(\boldsymbol{x})$:

$$\tilde{p}_\phi(\boldsymbol{x}) = \exp(E_\phi(\boldsymbol{x}))$$

Step 2: Enforcing Normalization. To make $\tilde{p}_\phi(\boldsymbol{x})$ a valid probability density, we must divide it by its integral over the entire space as follows:

$$p_\phi(\boldsymbol{x}) = \frac{\tilde{p}_\phi(\boldsymbol{x})}{\int \tilde{p}_\phi(\boldsymbol{x}')\,\mathrm{d}\boldsymbol{x}'} = \frac{\exp(E_\phi(\boldsymbol{x}))}{Z(\phi) := \int \exp(E_\phi(\boldsymbol{x}'))\,\mathrm{d}\boldsymbol{x}'},$$

where $Z(\phi)$ is known as the normalizing constant or partition function

$$Z(\phi) := \int \exp(E_\phi(\boldsymbol{x}'))\,\mathrm{d}\boldsymbol{x}'.$$

Challenges in modeling distributions

Step 1: Ensuring Non-Negativity. By applying a positive function to $E_\phi(\boldsymbol{x})$, such as $|E_\phi(\boldsymbol{x})|$, $E_\phi^2(\boldsymbol{x})$, or $\exp(E_\phi(\boldsymbol{x}))$. A standard and convenient choice is the exponential function. It gives us $\tilde{p}_\phi(\boldsymbol{x})$:

$$\tilde{p}_\phi(\boldsymbol{x}) = \exp(E_\phi(\boldsymbol{x}))$$

Step 2: Enforcing Normalization. To make $\tilde{p}_\phi(\boldsymbol{x})$ a valid probability density, we must divide it by its integral over the entire space as follows:

$$p_\phi(\boldsymbol{x}) = \frac{\tilde{p}_\phi(\boldsymbol{x})}{\int \tilde{p}_\phi(\boldsymbol{x}')\,\mathrm{d}\boldsymbol{x}'} = \frac{\exp(E_\phi(\boldsymbol{x}))}{Z(\phi) := \int \exp(E_\phi(\boldsymbol{x}'))\,\mathrm{d}\boldsymbol{x}'},$$

where $Z(\phi)$ is known as the normalizing constant or partition function

$$Z(\phi) := \int \exp(E_\phi(\boldsymbol{x}'))\,\mathrm{d}\boldsymbol{x}'.$$

The computation of $Z(\phi)$ is intractable. This is a central problem that motivates the development of many different families of DGMs.

Example 1: Energy-Based Models (EBMs)

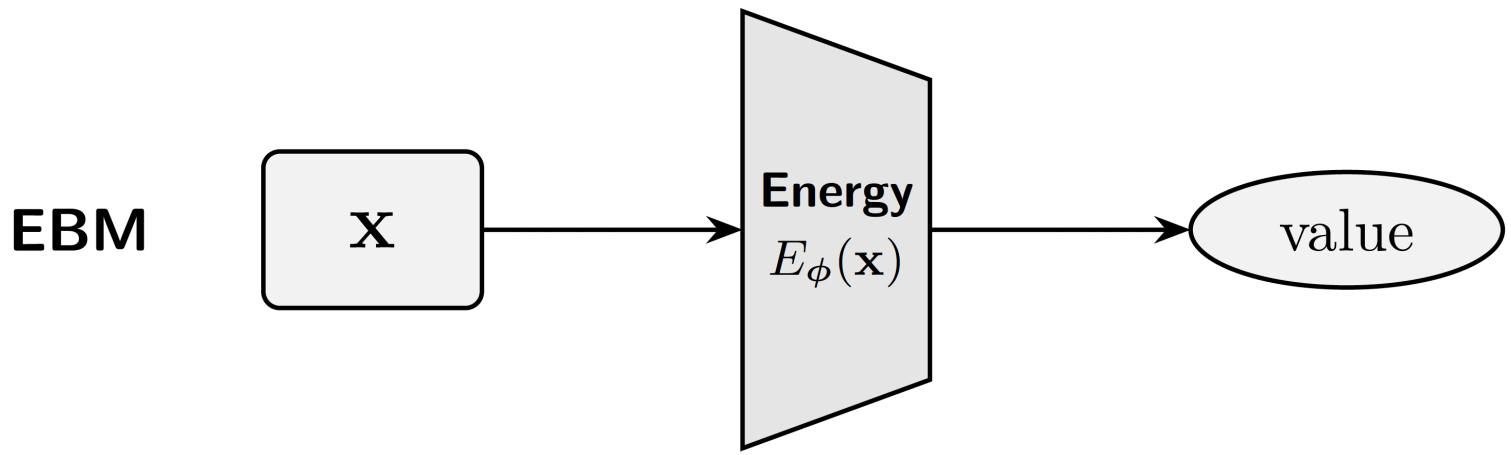


Figure 1.1: EBM maps an input $\boldsymbol{x}$ to a scalar energy

Example 1: Energy-Based Models (EBMs)

EBMs define $p_\phi(\boldsymbol{x})$ through an energy function $E_\phi(\boldsymbol{x})$ that *assigns lower energy to more probable data points*. The probability $p_\phi(\boldsymbol{x})$ is defined as:

$$p_\phi(\boldsymbol{x}) := \frac{\exp\left(-E_\phi(\boldsymbol{x})\right)}{Z(\phi)}, \text{ where } Z(\phi) = \int \exp\left(-E_\phi(\boldsymbol{x})\right) \mathrm{d}\boldsymbol{x}. \tag{1.6}$$

Example 1: Energy-Based Models (EBMs)

EBMs define $p_\phi(\boldsymbol{x})$ through an energy function $E_\phi(\boldsymbol{x})$ that *assigns lower energy to more probable data points*. The probability $p_\phi(\boldsymbol{x})$ is defined as:

$$p_\phi(\boldsymbol{x}) := \frac{\exp\left(-E_\phi(\boldsymbol{x})\right)}{Z(\phi)}, \text{ where } Z(\phi) = \int \exp\left(-E_\phi(\boldsymbol{x})\right) \mathrm{d}\boldsymbol{x}. \tag{1.6}$$

1. Ideas [Ackley et al., 1985, LeCun et al., 2006]
2. $p_\phi(\boldsymbol{x})$ is called Boltzmann distribution
3. $Z(\phi)$ is the partition function and computationally intractable
4. To avoid the calculation of $Z(\phi)$, diffusion models offer an alternative by generating data from the gradient of the log density, which does not depend on $Z(\phi)$. (More details in Lecture 03)
5. Training EBMs usually involves maximizing the MLE of the data

Example 1: Energy-Based Models (EBMs)

How to train your energy-based models [Song and Kingma, 2021]?

- The standard for learning p_ϕ from i.i.d. data is MLE. We fit $p_\phi(\boldsymbol{x})$ to $p_{\text{data}}(\boldsymbol{x})$ by maximizing the expected log-likelihood function over p_{data}, defined by

$$\arg\max_{\phi} \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})} [\log p_\phi(\boldsymbol{x})] \Leftrightarrow$$

$$\arg\min_{\phi} - \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})} [\log p_\phi(\boldsymbol{x})] \Leftrightarrow$$

$$\arg\min_{\phi} \left\{ \mathcal{D}_{\text{KL}} \left(p_{\text{data}}(\boldsymbol{x}) \| p_\phi(\boldsymbol{x}) \right) - \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})} [\log p_{\text{data}}(\boldsymbol{x})] \right\} \Leftrightarrow$$

$$\arg\min_{\phi} \mathcal{D}_{\text{KL}} \left(p_{\text{data}}(\boldsymbol{x}) \| p_\phi(\boldsymbol{x}) \right)$$

- We cannot directly compute the likelihood of an EBM, but we can still estimate the gradient of the log-likelihood with MCMC approaches.

Example 1: Energy-Based Models (EBMs)

Why gradient information? If two continuously differentiable real-valued functions $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ have equal first derivatives everywhere, then $f(\boldsymbol{x}) \equiv g(\boldsymbol{x}) + C$. When $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ are log probability density functions (PDFs) with equal first derivatives, and therefore $f(\boldsymbol{x}) \equiv g(\boldsymbol{x})$ (see Claim 2).

Example 1: Energy-Based Models (EBMs)

Why gradient information? If two continuously differentiable real-valued functions $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ have equal first derivatives everywhere, then $f(\boldsymbol{x}) \equiv g(\boldsymbol{x}) + C$. When $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ are log probability density functions (PDFs) with equal first derivatives, and therefore $f(\boldsymbol{x}) \equiv g(\boldsymbol{x})$ (see Claim 2).

The gradient of the log-probability of an EBM in Equ. (1.6) decomposes as a sum of two terms:

$$\nabla_\phi \log p_\phi(\boldsymbol{x}) = -\nabla_\phi E_\phi(\boldsymbol{x}) - \nabla_\phi \log Z(\phi) \tag{1.7}$$

- The first term, $-\nabla_\phi E_\phi(\boldsymbol{x})$, is straightforward to evaluate with NNs.
- The second term, $\nabla_\phi \log Z(\phi)$ is intractable to compute exactly.
- Score matching (Lecture 03):

$$\mathcal{L}_{\mathrm{SM}}(\phi) = \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\mathrm{data}}} \|\nabla_{\boldsymbol{x}} \log p_\phi(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} \log p_{\mathrm{data}}(\boldsymbol{x})\|_2^2 . \tag{1.8}$$

Example 2: Autoregressive Models

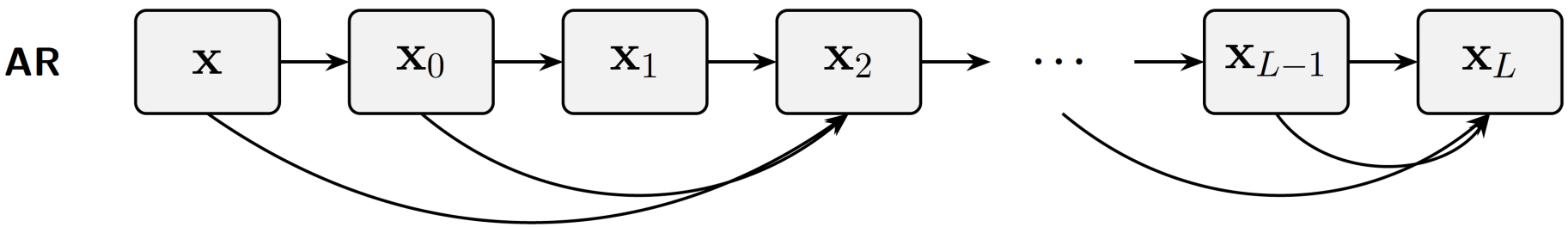


Figure 1.2: AR generates a sequence $\{\boldsymbol{x}_\ell\}$ left to right with causal dependencies

Example 2: Autoregressive Models

Deep autoregressive (AR) models factorize the joint data distribution p_ϕ into a product of conditional probabilities using the chain rule of probability:

$$p_\phi(\boldsymbol{x}) = \prod_{i=1}^{D} p_\phi\left(x_i \mid \boldsymbol{x}_{<i}\right), \tag{1.9}$$

where $\boldsymbol{x} = (x_1, \ldots, x_D)$ and $\boldsymbol{x}_{<i} = (x_1, \ldots, x_{i-1})$.

- Each conditional $p_\phi\left(x_i \mid \boldsymbol{x}_{<i}\right)$ is parameterized by a neural network
- Each $p_\phi\left(x_i \mid \boldsymbol{x}_{<i}\right)$ is normalized by design (e.g., via softmax for discrete or parameterized Gaussian for continuous variables)
- Training proceeds by maximizing the exact likelihood, or equivalently, minimizing the negative log-likelihood
- A foundational class of likelihood-based generative models and key approaches in modern research

Example 3: Variational Autoencoders (VAEs)

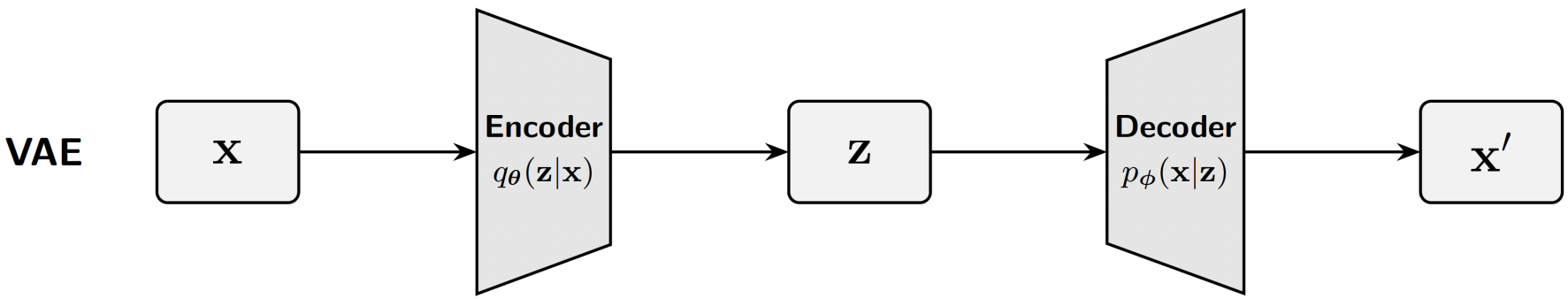


Figure 1.3: VAE encodes $\boldsymbol{x}$ to a latent $\boldsymbol{z}$ and decodes to a reconstruction $\boldsymbol{x}'$

Example 3: Variational Autoencoders (VAEs)

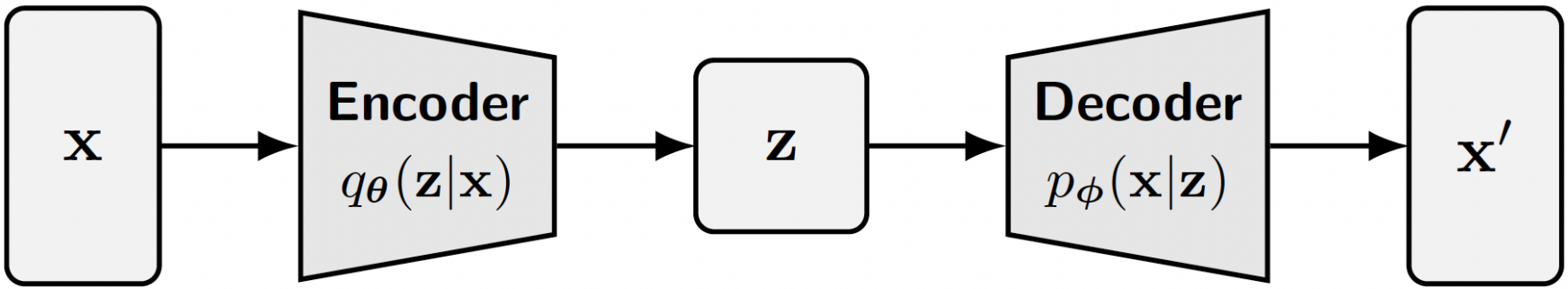


Figure 1.4: VAE consists of a stochastic encoder $q_{\theta}(\boldsymbol{z} \mid \boldsymbol{x})$ that maps data $\boldsymbol{x}$ to a latent variable $\boldsymbol{z}$, and a decoder $p_{\phi}(\boldsymbol{x} \mid \boldsymbol{z})$ that reconstructs data from the latent.

Example 3: Variational Autoencoders (VAEs)

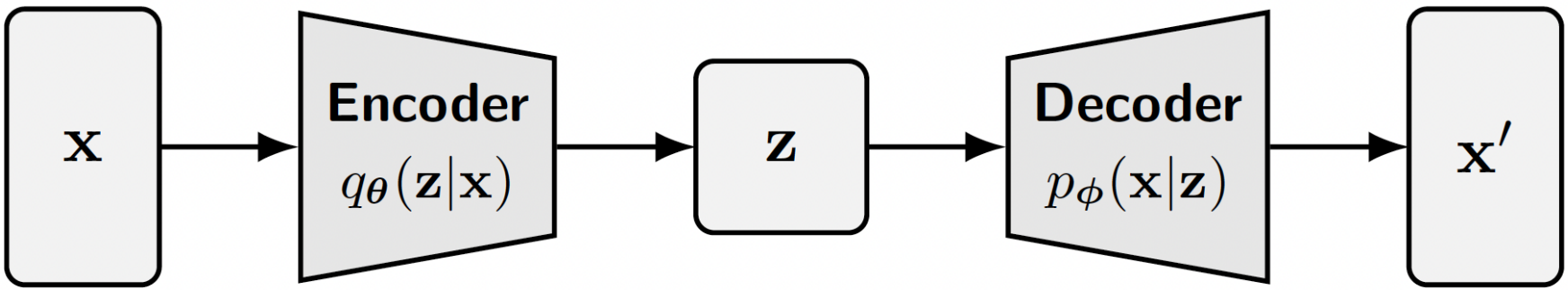


Figure 1.4: VAE consists of a stochastic encoder $q_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \boldsymbol{x})$ that maps data $\boldsymbol{x}$ to a latent variable $\boldsymbol{z}$, and a decoder $p_{\boldsymbol{\phi}}(\boldsymbol{x} \mid \boldsymbol{z})$ that reconstructs data from the latent.

- Ideas [Hinton and Salakhutdinov, 2006, Kingma and Welling, 2013]
- From a probabilistic view, VAEs learn both an encoder (inference network), $q_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \boldsymbol{x})$, which approximates the unknown distribution of latent variables given the data, and a decoder (the generator), $p_{\boldsymbol{\phi}}(\boldsymbol{x} \mid \boldsymbol{z})$, which reconstructs data from these latent variables.

Example 3: Variational Autoencoders (VAEs)

To define a computable training objective, since we cannot directly optimize $\log p_\phi(\boldsymbol{x})$, we can maximize a lower bound on it, that is, the Evidence Lower Bound (ELBO):

Example 3: Variational Autoencoders (VAEs)

To define a computable training objective, since we cannot directly optimize $\log p_\phi(\boldsymbol{x})$, we can maximize a lower bound on it, that is, the Evidence Lower Bound (ELBO):

- **Step 1:** The likelihood $p_\phi(\boldsymbol{x})$ can be treated as a latent-variable generative model through the marginal likelihood:

$$p_\phi(\boldsymbol{x}) = \int p_\phi(\boldsymbol{x} \mid \boldsymbol{z}) p(\boldsymbol{z}) \mathrm{d}\boldsymbol{z} \tag{1.10}$$

Example 3: Variational Autoencoders (VAEs)

To define a computable training objective, since we cannot directly optimize $\log p_\phi(\boldsymbol{x})$, we can maximize a lower bound on it, that is, the Evidence Lower Bound (ELBO):

- **Step 1:** The likelihood $p_\phi(\boldsymbol{x})$ can be treated as a latent-variable generative model through the marginal likelihood:

$$p_\phi(\boldsymbol{x}) = \int p_\phi(\boldsymbol{x} \mid \boldsymbol{z}) p(\boldsymbol{z}) \mathrm{d}\boldsymbol{z} \tag{1.10}$$

- **Step 2:** To connect our intractable generator to real data, consider the reverse question: given an observation $\boldsymbol{x}$, what latent codes $\boldsymbol{z}$ could have produced it? By Bayes' rule, the posterior distribution is

$$p_\phi(\boldsymbol{z} \mid \boldsymbol{x}) = \frac{p_\phi(\boldsymbol{x} \mid \boldsymbol{z}) p(\boldsymbol{z})}{p_\phi(\boldsymbol{x})} = \frac{p_\phi(\boldsymbol{x} \mid \boldsymbol{z}) p(\boldsymbol{z})}{\int_{\boldsymbol{z}'} p_\phi(\boldsymbol{x}, \boldsymbol{z}') \mathrm{d}\boldsymbol{z}'}. \tag{1.11}$$

  The denominator is intractable.

Example 3: Variational Autoencoders (VAEs)

- **Step 3:** The *variational* step in VAEs addresses Step 3 by replacing $p_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ with a tractable approximation $q_\theta(\boldsymbol{z} \mid \boldsymbol{x})$, which is an encoder (or inference network) parameterized by $\boldsymbol{\theta}$:

$$q_\theta(\boldsymbol{z} \mid \boldsymbol{x}) \approx p_\phi(\boldsymbol{z} \mid \boldsymbol{x}). \tag{1.12}$$

  In practice, the encoder maps each observed data point $\boldsymbol{x}$ to a distribution over latent codes.

Example 3: Variational Autoencoders (VAEs)

- **Step 3:** The *variational* step in VAEs addresses Step 3 by replacing $p_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ with a tractable approximation $q_\theta(\boldsymbol{z} \mid \boldsymbol{x})$, which is an encoder (or inference network) parameterized by $\boldsymbol{\theta}$:

$$q_\theta(\boldsymbol{z} \mid \boldsymbol{x}) \approx p_\phi(\boldsymbol{z} \mid \boldsymbol{x}). \tag{1.12}$$

  In practice, the encoder maps each observed data point $\boldsymbol{x}$ to a distribution over latent codes.
- **Step 4:** The log likelihood can then be treated as marginal of the density function $p_\phi(\boldsymbol{x}, \boldsymbol{z})$ over $\boldsymbol{z}$

$$\begin{aligned}\log p_\phi(\boldsymbol{x}) &= \log \int p_\phi(\boldsymbol{x}, \boldsymbol{z}) \mathrm{d}\boldsymbol{z} = \log \int q_\theta(\boldsymbol{z} \mid \boldsymbol{x}) \frac{p_\phi(\boldsymbol{x}, \boldsymbol{z})}{q_\theta(\boldsymbol{z} \mid \boldsymbol{x})} \mathrm{d}\boldsymbol{z} \\ &= \log \mathbb{E}_{\boldsymbol{z} \sim q_\theta(\boldsymbol{z}|\boldsymbol{x})} \left[\frac{p_\phi(\boldsymbol{x}, \boldsymbol{z})}{q_\theta(\boldsymbol{z} \mid \boldsymbol{x})} \right] \geq \underbrace{\mathbb{E}_{\boldsymbol{z} \sim q_\theta(\boldsymbol{z}|\boldsymbol{x})} \left[\log \frac{p_\phi(\boldsymbol{x}, \boldsymbol{z})}{q_\theta(\boldsymbol{z} \mid \boldsymbol{x})} \right]}_{\text{Evidence Lower Bound (ELBO)}},\end{aligned}$$

  where the inequality is due to Jensen's inequality in Claim 3.

Example 3: Variational Autoencoders (VAEs)

Theorem 1 (Evidence Lower Bound (ELBO))

For any data point $\boldsymbol{x}$, the log-likelihood satisfies:

$$\log p_{\boldsymbol{\phi}}(\boldsymbol{x}) \geq \mathcal{L}_{\mathrm{ELBO}}(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{\phi}), \tag{1.13}$$

where the ELBO is given by

$$\mathcal{L}_{\mathrm{ELBO}}(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{\phi}) = \underbrace{\mathbb{E}_{\boldsymbol{z} \sim q_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})} \left[\log p_{\boldsymbol{\phi}}(\boldsymbol{x} \mid \boldsymbol{z})\right]}_{\textit{Reconstruction Term}} - \underbrace{\mathcal{D}_{\mathrm{KL}} \left(q_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \boldsymbol{x}) \| p(\boldsymbol{z})\right)}_{\textit{Latent Regularization}}$$

Example 3: Variational Autoencoders (VAEs)

Theorem 1 (Evidence Lower Bound (ELBO))

For any data point $\boldsymbol{x}$, the log-likelihood satisfies:

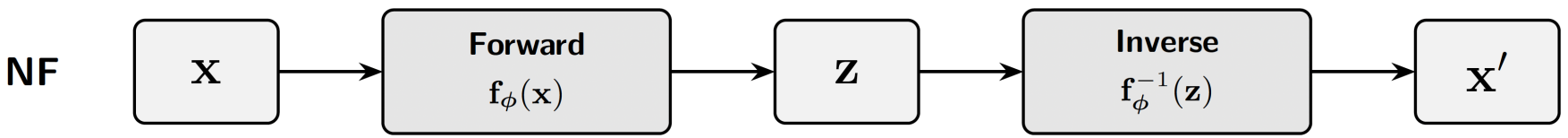
$$\log p_{\boldsymbol{\phi}}(\boldsymbol{x}) \geq \mathcal{L}_{\mathrm{ELBO}}(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{\phi}), \tag{1.13}$$

where the ELBO is given by

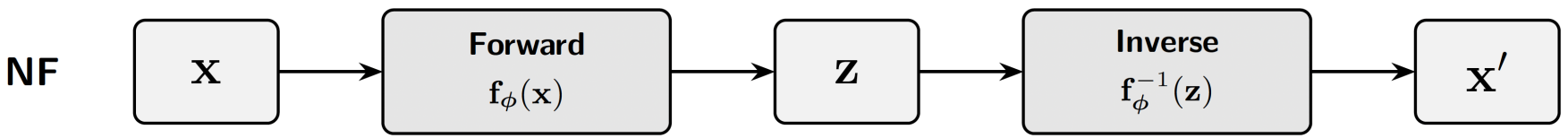
$$\mathcal{L}_{\mathrm{ELBO}}(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{\phi}) = \underbrace{\mathbb{E}_{\boldsymbol{z} \sim q_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})} \left[\log p_{\boldsymbol{\phi}}(\boldsymbol{x} \mid \boldsymbol{z})\right]}_{\textit{Reconstruction Term}} - \underbrace{\mathcal{D}_{\mathrm{KL}} \left(q_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \boldsymbol{x}) \| p(\boldsymbol{z})\right)}_{\textit{Latent Regularization}}$$

- The first term encourages accurate reconstruction of the data
- The second regularizes the latent variables by keeping them close to a simple prior distribution $p_{\text{prior}}(\boldsymbol{z}) := p(\boldsymbol{z})$ (often Gaussian)
- VAEs provide a principled way to combine neural networks with latent variable models and remain one of the most widely used approaches

Example 4: Normalizing Flows



Example 4: Normalizing Flows



- NF first applies an invertible map $\boldsymbol{f}_\phi$ between $\boldsymbol{x}$ and $\boldsymbol{z}$
- It then uses $\boldsymbol{f}_\phi^{-1}$ to produce $\boldsymbol{x}'$

Example 4: Normalizing Flows

- Classic flow-based models, such as Normalizing Flows (NFs) [Rezende and Mohamed, 2015] and Neural Ordinary Differential Equations (NODEs) [Chen et al., 2018] aim to learn a bijective mapping $\boldsymbol{f}_\phi$ between a simple latent distribution $\boldsymbol{z}$ and a complex data distribution $\boldsymbol{x}$ via an invertible operator.
- This is achieved either through a sequence of bijective transformations (in NFs) or by modeling the transformation as an Ordinary Differential Equation (in NODEs).
- These models leverage the "change-of-variable formula for densities", enabling MLE training:

$$\log p_\phi(\boldsymbol{x}) = \log p(\boldsymbol{z}) + \log \left| \det \frac{\partial \mathrm{f}_\phi^{-1}(\boldsymbol{x})}{\partial \boldsymbol{x}} \right| \tag{1.14}$$

Example 5: Generative Adversarial Networks

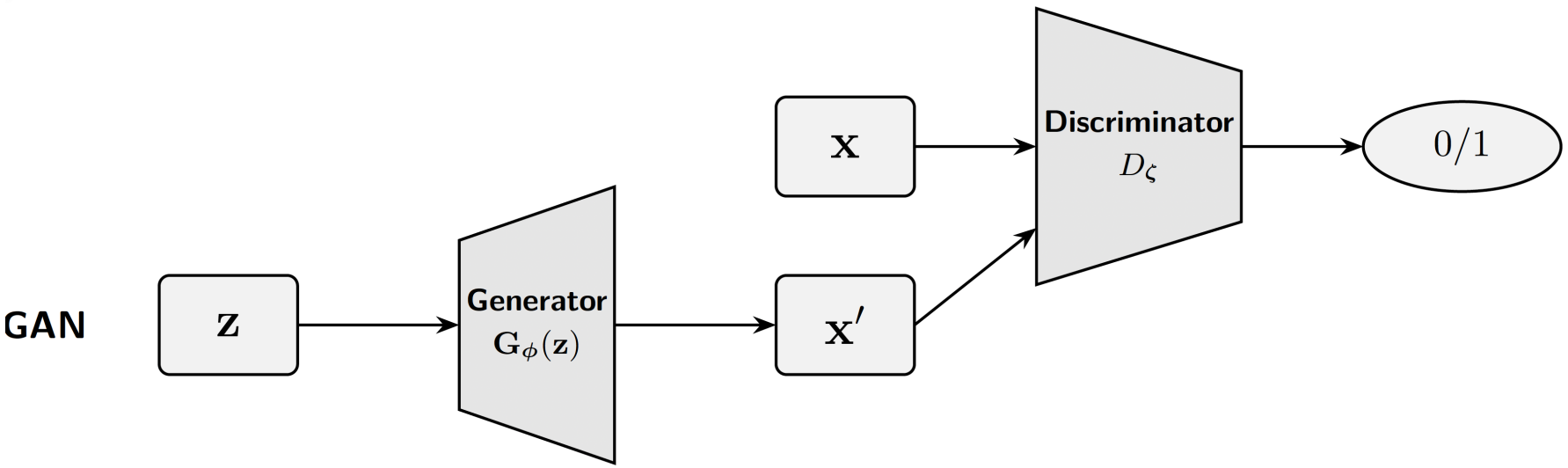


Figure 1.5: GAN transforms noise $\boldsymbol{z}$ to a sample $\boldsymbol{x}'$ that is judged against real $\boldsymbol{x}$ by a discriminator D_ζ

Example 5: Generative Adversarial Networks

- Generative Adversarial Networks (GANs) consist of two NNs, a generator G_ϕ and a discriminator D_ζ, that compete against each other.
- The generator G_ϕ aims to create realistic samples $G_\phi(\boldsymbol{z})$ from random noise $\boldsymbol{z} \sim p_{\text{prior}}$, while the discriminator attempts to distinguish between real samples $\boldsymbol{x}$ and generated samples $\mathrm{G}_\phi(\boldsymbol{z})$.
- The objective function for GANs can be formulated as:

$$\min_{G_\phi} \max_{D_\zeta} \underbrace{\mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} \left[\log D_\zeta(\boldsymbol{x})\right]}_{\text{real}} + \underbrace{\mathbb{E}_{\boldsymbol{z} \sim p_{\text{prior}}(\boldsymbol{z})} \left[\log\left(1 - D_\zeta\left(G_\phi(\boldsymbol{z})\right)\right)\right]}_{\text{fake}}.$$

- GANs do not define an explicit density function and therefore bypass likelihood estimation entirely. They focus on generating samples that closely mimic the data distribution.

Example 5: Generative Adversarial Networks

From a divergence perspective, the discriminator D_ζ implicitly measures the discrepancy between the true data distribution p_{data} and the generator distribution p_{G_ϕ}, where p_{G_ϕ} denotes the distribution of generated samples $\mathsf{G}_\phi(\boldsymbol{z})$ obtained from noise $\boldsymbol{z} \sim p_{\text{prior}}$. With an optimal discriminator for a fixed generator G_ϕ computed as

$$\frac{p_{\text{data}}\left(\boldsymbol{x}\right)}{p_{\text{data}}\left(\boldsymbol{x}\right) + p_{\mathsf{G}_\phi}(\boldsymbol{x})}, \tag{1.15}$$

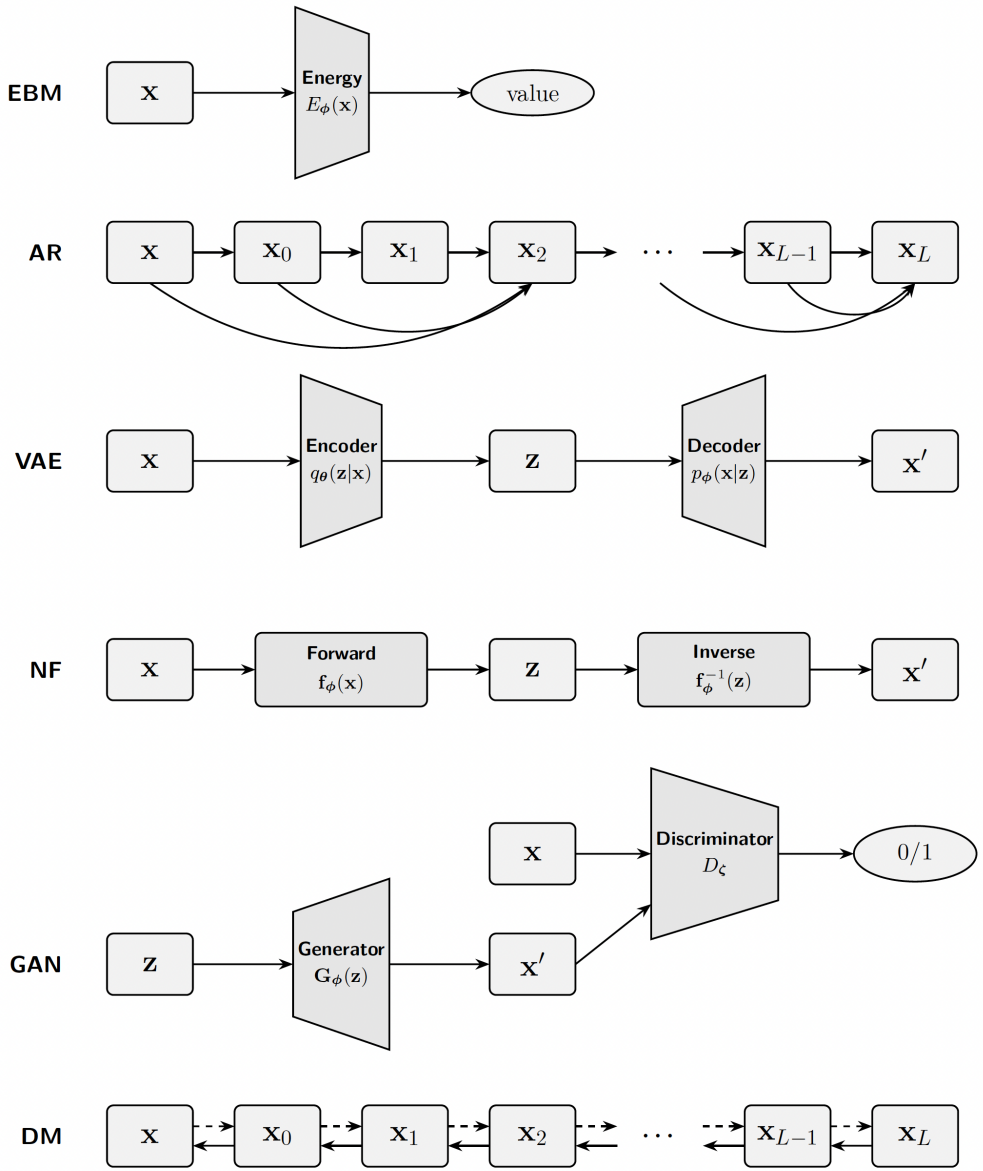
the generator's minimization reduces to

$$\min_{\mathsf{G}_\phi} 2\mathcal{D}_{\text{JS}}\left(p_{\text{data}} \,\|p_{\mathsf{G}_\phi}\right) - \log 4. \tag{1.16}$$

Here, $\mathcal{D}_{\text{JS}}$ denotes the Jensen-Shannon divergence, defined as

$$\mathcal{D}_{\text{JS}}(p\|q) := \frac{1}{2}\mathcal{D}_{\text{KL}}\left(p\|\frac{p+q}{2}\right) + \frac{1}{2}\mathcal{D}_{\text{KL}}\left(q\|\frac{p+q}{2}\right). \tag{1.17}$$

Prominent DGMs



Quick Summary of DGMs

DGMs span a wide spectrum of modeling strategies. A fundamental distinction lies in how these models parameterize the underlying data distribution, that is, whether they specify $p_\phi(\boldsymbol{x})$ explicitly or only implicitly, irrespective of the training objective.

- **Explicit Models**: These models directly parameterize a probability distribution $p_\phi(\boldsymbol{x})$ via a tractable or approximately tractable density or mass function. Examples include ARs, NFs, VAEs, and DMs, all of which define $p_\phi(\boldsymbol{x})$ either exactly or through a tractable bound.
- **Implicit Models**: These models specify a distribution only through a sampling procedure, typically of the form $\boldsymbol{x} = G_\phi(\boldsymbol{z})$ for some noise variable $\boldsymbol{z} \sim p_{\text{prior}}$. In this case, $p_\phi(\boldsymbol{x})$ is not available in closed form and may not be defined at all.

Closing Remarks

- We begin by defining the primary objective: to learn a tractable model distribution p_ϕ parametrized by ϕ that approximates an unknown, complex data distribution p_{data}.
- A central challenge is the computational intractability of the normalizing constant, or partition function $Z(\phi)$.
- To circumvent this problem, various families of DGMs have been developed, each employing a distinct strategy. We surveyed several prominent approaches, including EBMs, ARs, VAEs, NFs, and GANs.
- While each of these classical frameworks is significant, three in particular serve as the conceptual origins for the diffusion models that are the focus of this monograph: VAEs, EBMs, and NFs. We will trace the evolution of diffusion models from these three foundational paradigms.

Claim 1 (Equal derivatives means differences by constant)

If two continuous functions f and g have the same derivatives in the same domain, then $f(\boldsymbol{x}) = g(\boldsymbol{x}) + C$ where C is a constant.

Proof.

Let $f : \mathbb{R}^d \to \mathbb{R}$ and $g : \mathbb{R}^d \to \mathbb{R}$ be continuously differentiable and suppose

$$\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} g(\boldsymbol{x}) \quad \text{for all } \boldsymbol{x}.$$

Define $h(\boldsymbol{x}) = f(\boldsymbol{x}) - g(\boldsymbol{x})$. Then

$$\nabla_{\boldsymbol{x}} h(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} f(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} g(\boldsymbol{x}) = 0.$$

That is, $h(\boldsymbol{x})$ must be constant. This leads to

$$h(\boldsymbol{x}) \equiv C \quad \Rightarrow \quad f(\boldsymbol{x}) = g(\boldsymbol{x}) + C.$$

■

Claim 2 (Equal derivatives mean equal functions)

Suppose f and g are log densities: $p(\boldsymbol{x}) = e^{f(\boldsymbol{x})}$, $q(\boldsymbol{x}) = e^{g(\boldsymbol{x})}$, and both are normalized PDFs, i.e., $\int e^{f(\boldsymbol{x})}\, d\boldsymbol{x} = 1$, $\int e^{g(\boldsymbol{x})}\, d\boldsymbol{x} = 1$. If the gradients of f and g are equal everywhere, then $f \equiv g$.

Proof.

If the gradients of f and g are equal everywhere, by Claim 1, $f(\boldsymbol{x}) = g(\boldsymbol{x}) + C$. This means $e^{f(\boldsymbol{x})} = e^{g(\boldsymbol{x})+C} = e^C\, e^{g(\boldsymbol{x})}$. Integrate both sides:

$$1 = \int e^{f(\boldsymbol{x})} d\boldsymbol{x} = \int e^C e^{g(\boldsymbol{x})}\, d\boldsymbol{x} = e^C \int e^{g(\boldsymbol{x})}\, d\boldsymbol{x} = e^C \cdot 1 = e^C.$$

The only choice is $C = 0$. Hence $f(\boldsymbol{x}) = g(\boldsymbol{x})$ for all $\boldsymbol{x}$, and therefore $p(\boldsymbol{x}) = q(\boldsymbol{x})$. ■

Jensen's Inequality

Jensen's inequality generalizes the statement that the secant line of a convex function lies above the graph of the function: the secant line consists of weighted means of the convex function (for $t \in [0, 1]$).

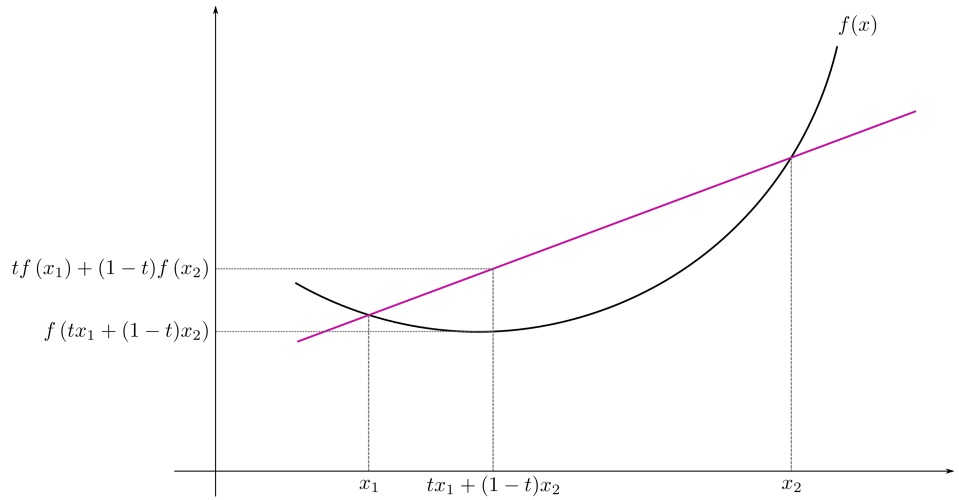


Figure 1.6: A secant line of a convex function lies above its graph.

- Weighted means of the convex function: $tf(x_1) + (1-t)f(x_2)$
- The convex function of the weighted means: $f(tx_1 + (1-t)x_2)$

In this case, Jensen's inequality provides

$$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2).$$

Jensen's Inequality

- **Finite form**: For a real convex function φ, numbers $x_1, x_2, \ldots, x_n$ in its domain, and positive weights a_i, Jensen's inequality can be stated as:

$$\varphi\left(\frac{\sum a_i x_i}{\sum a_i}\right) \leq \frac{\sum a_i \varphi\left(x_i\right)}{\sum a_i}$$

  and the inequality is reversed if φ is concave, which is

$$\varphi\left(\frac{\sum a_i x_i}{\sum a_i}\right) \geq \frac{\sum a_i \varphi\left(x_i\right)}{\sum a_i}.$$

  Equality holds if and only if $x_1 = x_2 = \cdots = x_n$ or φ is linear on a domain containing $x_1, x_2, \cdots, x_n$.

- **Example**: The function $\log(x)$ is concave, we have the following inequality

$$\log\left(\frac{\sum_{i=1}^{n} x_i}{n}\right) \geq \frac{\sum_{i=1}^{n} \log\left(x_i\right)}{n}$$

Jensen's Inequality

Claim 3 (Probabilistic form of Jensen's Inequality)

In the probability theory setting, let (Ω, F, P) *be a probability space,* X *an integrable real-valued random variable and* φ *a convex function. Then*

$$\varphi(\mathbb{E}[X]) \leq \mathbb{E}[\varphi(X)]. \tag{1.18}$$

- Note that the equality holds if and only if φ is a linear function on some convex set A such that $P(X \in A) = 1$.

David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann machines. *Cognitive science*, 9(1):147–169, 1985.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, Fujie Huang, et al. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Frank Nielsen. Two tales for a geometric jensen–shannon divergence. *arXiv preprint arXiv:2508.05066*, 2025.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. pmlr, 2015.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Lan Tao, Shirong Xu, Chi-Hua Wang, Namjoon Suh, and Guang Cheng. Discriminative estimation of total variation distance: A fidelity auditor for generative data. *arXiv preprint arXiv:2405.15337*, 2024.