The Foundations of Generative AI

Indrajit Bhattacharya
KnowDis AI
indrajitb@gmail.com | indrajit.bhattacharya@knowdis.ai

Tutorial at CODS 2025
IISER, Pune
December 17, 2025

Abstract

This 'tutorial' is more of a guided tour exploring the foundations of generative AI from many different points of view, with the goal of dispelling prevailing 'gen AI'-related myths and misunderstandings.

We will begin by defining generative AI, understanding what distinguishes it from its 'non-generative' counterpart (yes, that exists!) and appreciating how generative AI approaches the essence of human and super-human agency.

We will then define the core tasks in generative AI, namely, sampling, scoring, inference and learning, in addition to the meta-task of modeling. We will analyze the key challenges involved in each of these, and the intrinsic trade-offs involved in addressing these simultaneously.

We will then turn our gaze backwards in time and look far beyond 2012 (yes, AI did exist back then!), beyond even the 1950s, to appreciate the debt that contemporary researchers and developers owe to giants from different cultures, continents, disciplines and eras for the tools and techniques used to address these core tasks.

Next, we will review the building blocks of generative AI to understand how cutting-edge generative AI models of today borrow from and build upon traditional generative models (no, I don't mean GPT-1!). Specifically, we will review generative models for classification and regression, IID versus structured models, conditional independence and the graphical model view, latent variables (mixture models versus factor analysis), and the troublesome partition function.

Then, and only then (yes, we are taking the long route!), we will review a few specific categories of generative models — autoregressive models, variational autoencoders, generative adversarial networks, normalizing flows, and diffusion models — and analyze how these arrive at different compromises between the competing goals of generative modeling.

We will end by thinking through the significant risks posed by generative AI alongside transformative benefits, and discuss the deep ethical considerations involved in building, owning and using generative AI models.

"What do they know of cricket who only cricket know."
— C.L.R. James

What do they know of AI, who only (open)AI know.

Contents

Part I

Foundations of Generative AI

1 Introduction

1.1 A Familiar Experience for AI Practitioners

The Temperature Parameter: A Window into Generative AI

- You've used ChatGPT's temperature parameter. Most scientists set it to zero for deterministic outputs. But why does it exist at all?
- **Core insight**: This parameter reveals what generative models fundamentally do—they represent and sample from *distributions* over possibilities, not single predictions.
- Temperature controls distribution shape:
  - Temperature = 0: "Give me your most confident prediction"
  - Higher temperature: "Show me the range of possibilities"
- **Discriminative vs. Generative**:
  - Discriminative: "What is this?" → single classification
  - Generative: "What could this be?" → sample from p(data|condition)
- **Historical debt**: Temperature comes from statistical mechanics and Boltzmann distributions (1800s particle physics). The same mathematics describing gas molecules now controls language model creativity.
- Understanding this seemingly simple parameter unlocks the entire framework of generative modeling.

1.2 Examples from the Sciences and the Arts

Common structure across domains:

- **Molecular Design**: Navigate 10^{60} possible drug-like molecules. Constraints: valence rules, synthesizability, non-toxicity. Learn from: known drugs, structure-activity relationships.
- **Experimental Protocol**: Choose among many valid experimental designs. Constraints: equipment availability, statistical soundness, hypothesis relevance. Learn from: methods sections, prior experiments, failures.
- **Device Design**: Arrange materials in countless geometric configurations. Constraints: Maxwell's equations, fabrication limits, performance targets. Learn from: materials databases, simulations, testing.
- **Novel Writing**: Infinite narrative possibilities (characters, plots, settings). Constraints: internal consistency, coherent progression, emotional engagement. Learn from: reading, feedback, resonance.
- **Poetry**: Endless word/rhythm/image combinations. Constraints: semantic coherence, formal structure (optional), aesthetic impact. Learn from: reading poetry, experimentation, sensing what works.

1.3 The Pattern: Three Essential Elements

1. **Vast possibility space**: 10^{60} molecules, infinite narratives, endless word combinations
2. **Constraints**: Hard (syntax, physics, chemistry) and soft (aesthetics, function, coherence)
3. **Learning from examples**: No one starts from scratch—expertise comes from data, feedback, accumulated experience

This is the essence of generative AI: Creating artifacts in vast possibility spaces, respecting constraints, by learning patterns from data.

1.4 Discriminative AI: A Special Case

What if the output space collapsed?

- Not molecules or stories, but a handful of labels: "Diseased/healthy," "Spam/not spam," "Cat/dog/bird"
- This is **discriminative AI**—predicting labels from inputs
- Still involves constraints and learning from examples, but possibility space has collapsed from infinite to a few discrete choices
- Tremendously successful, but a special case of a more general framework

Two Key Insights:

1. **Generative models are more fundamental**: If you have p(data, labels), derive discriminative predictions via Bayes rule. Generation → discrimination works; the reverse doesn't.
2. **Complex generation = composed conditionals**: Generate molecule atom-by-atom, story word-by-word. Simple conditional predictions compose into rich structures.

The unified view: Generative AI provides a probabilistic framework where discrimination is one application, and complex creation emerges from learned conditional distributions.

1.5 Why Generative AI?

We model full probability distributions $P_\theta(\mathbf{x})$ over data. Why this approach? What does it enable?

1.5.1 Generation and Decision: A Fundamental Relationship

- **Discriminative AI**: Outputs $P(y|\mathbf{x})$ over small label sets → collapse to single prediction (argmax, thresholding) → returns one distribution atom
- **Generative AI**: Outputs distributions over high-dimensional spaces → sample multiple instances → explores possibilities
- **Key insight**: Decision-making presupposes a space of possibilities. To choose, there must be options. But where do these options come from?

1.5.2 The Primacy of Generation

Generation is ontologically prior to decision.

- Every choice set was generated by some process: designers, evolution, physical processes, previous creation
- Discriminative AI operates on outputs of generative processes (human or natural)
- You cannot decide without options; generation produces options
- Generation doesn't just navigate possibility spaces—it creates them

1.5.3 Why Distributions and Variance Matter

Variance is the wellspring of creation.

- Distributions capture what is *possible*, not just what is most probable
- Variance represents diversity of the possibility space
- Without variance: only one outcome, no choices, nothing to decide among
- Embracing variance enables:
  - Exploration of multiple plausible solutions
  - Creative synthesis of novel artifacts
  - Discovery beyond what we've seen
  - Diversity in valid answers
  - Uncertainty quantification
- Discriminative AI minimizes variance (confident predictions). Generative AI embraces variance as creative source.

1.5.4 What Generative AI Enables

- **Creation and Design**: Generate solutions that don't exist yet—molecules, materials, protocols. Possibility space too vast to enumerate.
- **Understanding Through Synthesis**: "What I cannot create, I do not understand" (Feynman). Generation tests understanding beyond recognition.
- **Hypothesis Generation**: Propose new theories, experiments, explanations. These must be generated, not selected.
- **Reasoning Under Uncertainty**: Full distributions capture ambiguity; multiple samples show different plausible paths.
- **Expanding Possibility Spaces**: Create new options, not just navigate existing ones.

1.5.5 Connection to Opening Examples

Our examples (molecules, protocols, poetry) are *not* **selection problems**:

- Chemist generates candidates in vast space, doesn't enumerate 10^{60} molecules
- Poet creates new verse, doesn't select from existing poems
- Engineer designs novel structures, doesn't pick from catalog

These are generative tasks requiring creation of possibilities, not decisions among them. They require distributions—variance over structures, narratives, expressions—because **variance makes creation possible**.

Generative AI provides the mathematical framework for this fundamental capability: bringing new possibilities into existence.

1.6 The Mathematical Framework

- **Data (x)**: Feature vectors, images, video, text, graphs, point clouds, audio, etc.
- **Labels (y)**: Binary values, categorical classes, integers, real numbers, vectors, text sequences, images, etc.

Generative vs Discriminative/Prescriptive Tasks Machine learning tasks can be categorized based on their input-output relationship:

- **Prescriptive (Discriminative) Tasks**: Given data x, output label y
  - Classification: Image → category
  - Regression: Features → continuous value
  - Object detection: Image → bounding boxes + labels
  - Semantic segmentation: Image → pixel-wise labels
- **Generative Tasks**: Generate data samples
  - **Unconditional generation**: Output data $x_1, x_2, \ldots, x_n$
    - Generate images from random noise
    - Generate text continuations
    - Generate molecular structures
  - **Conditional generation**: Output data $x_{m+1}, \ldots, x_{m+n}$ given prompt/context $x_1, \ldots, x_m$
    - Text given text (language modeling, machine translation)
    - Image given text (text-to-image synthesis)
    - Audio given text (text-to-speech)
    - Image given image (style transfer, super-resolution, inpainting)
    - Video given text (text-to-video)
    - 3D model given image (image-to-3D)
    - Code given text (code generation)

1.6.1 Probabilistic Framework for Generative AI

Generative AI is framed firmly in terms of **probability theory**:

Core Probabilistic Setup

1. **Data as random variable**: Data X is modeled as a random variable (or random vector)
2. **Unknown distribution**: X is associated with an unknown probability distribution P_θ, parameterized by θ
3. **Observed samples**: We observe a dataset $\mathcal{D} = \{x^{(1)}, x^{(2)}, \ldots, x^{(n)}\}$ consisting of samples from this distribution:
$$x^{(i)} \sim P_\theta(X) \quad \text{for } i = 1, \ldots, n \tag{1}$$
4. **IID assumption**: (Often) the samples are assumed to be **independently and identically distributed**:
$$P(x^{(1)}, \ldots, x^{(n)}|\theta) = \prod_{i=1}^{n} P_\theta(x^{(i)}) \tag{2}$$

We observe data $\mathbf{x} \in \mathcal{X}$ (molecules, images, text) drawn from some unknown empirical distribution $P_{\text{data}}(\mathbf{x})$.

Generative modeling aims to approximate $P_{\text{data}}(\mathbf{x})$ with a parameterized family of distributions $P_\theta(\mathbf{x})$, where θ are learnable parameters.

The Meta-Task - Modeling: Choose the model family—the functional form of $P_\theta(\mathbf{x})$ and its parameterization. This determines what distributions we can represent.

1.6.2 The Four Core Tasks

Given a model family $P_\theta(\mathbf{x})$:

1. Scoring: Evaluate the probability density $P_\theta(\mathbf{x})$ for a given data point $\mathbf{x}$. How likely is this molecule under our model?

2. Sampling: Generate new samples $\mathbf{x} \sim P_\theta(\mathbf{x})$. Create a new molecule from the learned distribution.

3. Learning: Estimate parameters θ from data $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ such that $P_\theta(\mathbf{x})$ matches $P_{\text{data}}(\mathbf{x})$. Make the model distribution close to the data distribution.

4. Inference: For models with latent variables $\mathbf{z}$, compute posterior distributions. If $\mathbf{x} = (\mathbf{x}_{\text{obs}}, \mathbf{x}_{\text{hidden}})$, infer $P_\theta(\mathbf{x}_{\text{hidden}} \mid \mathbf{x}_{\text{obs}})$. Encoding—mapping data to latent representations—is a special case.

Inference takes several forms:

- **Representation inference**: $P_\theta(\mathbf{z}|\mathbf{x})$ for latent variables. Continuous latents enable encoding, discrete latents enable clustering.
- **Reconstruction inference**: $P_\theta(\mathbf{x}_{\text{missing}}|\mathbf{x}_{\text{observed}})$ for partial observations. Includes missing data imputation, denoising, and inverse problems.
- **Reasoning inference**: Multi-step logical inference, planning, counterfactual reasoning, question answering.

1.7 Conditional Generation

In practice, we often generate conditioned on some input or context $\mathbf{y}$. Instead of modeling $P_\theta(\mathbf{x})$, we model the conditional distribution $P_\theta(\mathbf{x} \mid \mathbf{y})$.

Examples:

- Generate a molecule $\mathbf{x}$ with desired properties $\mathbf{y}$ (binding affinity, solubility)

- Generate a story $\mathbf{x}$ from a prompt $\mathbf{y}$
- Generate an image $\mathbf{x}$ from a text description $\mathbf{y}$
- Design a device $\mathbf{x}$ meeting specifications $\mathbf{y}$

The four core tasks apply identically to conditional models:

- **Scoring**: Evaluate $P_\theta(\mathbf{x} \mid \mathbf{y})$
- **Sampling**: Generate $\mathbf{x} \sim P_\theta(\mathbf{x} \mid \mathbf{y})$
- **Learning**: Match $P_\theta(\mathbf{x} \mid \mathbf{y})$ to $P_{\text{data}}(\mathbf{x} \mid \mathbf{y})$
- **Inference**: Compute $P_\theta(\mathbf{z} \mid \mathbf{x}, \mathbf{y})$ for latent variables $\mathbf{z}$

Conditional generation is the norm rather than the exception in applications.

1.8 Goals and Pedagogical Approach

1.8.1 Understanding Through Construction: A Paradigm

Before we embark on our exploration of generative models, let us establish how we will build understanding throughout this tutorial. Rather than presenting the approach abstractly, we demonstrate it through a familiar example: the network of canonical probability distributions.

- **Bernoulli** (Jacob Bernoulli, *Ars Conjectandi*, 1713): simplest — binary outcome, parameter p
  - Question: model n trials? $\xrightarrow{\text{sum } n \text{ i.i.d.}}$ **Binomial**(n, p)
- **Branching**: Why only 2 outcomes?
  - Bernoulli $\xrightarrow{k \text{ outcomes}}$ **Categorical**; Binomial $\xrightarrow{k \text{ outcomes}}$ **Multinomial**
- **Different question**: What about processes — events unfolding over time?
  - Counting events:
    - Fixed n discrete trials $\to$ **Binomial**(n, p) (already seen)
    - Continuous time, rare events $\to$ **Poisson**(λ) (Siméon Denis Poisson, 1837)
  - Waiting time until first success:
    - Discrete trials $\to$ **Geometric**
    - Continuous time $\to$ **Exponential**(λ)
    - (Both are *memoryless* — the Markov property, Andrey Markov, 1906)
  - Until k-th success? **Negative Binomial** (discrete), **Gamma**(k, λ) (Euler, 1729)
- **Creating the dice**: So far we've been *given* parameter p. What if we want to model our uncertainty about p itself?
  - Need to assign probabilities to different values of $p \in [0, 1]$
  - But p is continuous! Can we reuse the exponential idea?
- **Extending exponential to model continuous values**:
  - **Key insight from Geometric/Exponential**: When we have infinite possibilities (countable or uncountable), decay is *inevitable* — without it, probabilities don't sum/integrate to 1

  - **Exponential** models $x \in [0, \infty)$ with decay: $p(x) \propto \exp(-\lambda x)$ (linear in exponent)
  - **Question**: What if we want to model values over all of $\mathbb{R}$ (not just positive)?
  - **Answer**: Need decay on *both sides* of some center μ
    * Simplest smooth symmetric decay: quadratic $(x - \mu)^2$
    * This gives: $p(x) \propto \exp(-(x - \mu)^2/2\sigma^2) \rightarrow$ **Normal**(μ, σ^2) (de Moivre 1733, Gauss 1809, Laplace 1812)
    * Properties: smooth, tractable, two parameters (location, scale)
    * (Maximum entropy given mean and variance — principled justification)
  - **CLT reveals universality**: Sums of i.i.d. random variables converge to Normal. Rigorous but "mystical" — accept it mathematically.

- **Back to modeling** p: Can we use Normal for $p \in [0, 1]$?
  - **Problem**: Normal assigns probability to negative values — nonsense for p!
  - **Two philosophical approaches**:

- **Solution 1 — Fix what we have**: Transform Normal to $[0, 1]$
  - Work in unconstrained space, then map to $[0, 1]$
  - **Logistic-Normal**: $p = \text{logistic}(z)$ where $z \sim \mathcal{N}(\mu, \sigma^2)$
  - Advantages: unconstrained optimization, neural-network friendly

- **Solution 2 — Redesign from scratch**: Build directly on $[0, 1]$
  - For Bernoulli: **Beta**(α, β) (Euler, 1729) — natural on $[0, 1]$, flexible
  - For Categorical: **Dirichlet**$(\alpha_1, \ldots, \alpha_k)$ (Dirichlet, 1839) — natural on simplex
  - Advantages: conjugate priors, interpretable parameters

- **Modeling choice**: Neither approach is "correct"
  - Beta/Dirichlet: conjugate, interpretable, domain-matched
  - Logistic-Normal: unconstrained, flexible, neural-friendly
  - Different priorities, both valid

Principles Demonstrated:

1. Start simple (Bernoulli)
2. Questions drive evolution
3. Operations create paths (sum, limit, transform, hierarchy)
4. Parallel structures (discrete $\leftrightarrow$ continuous)
5. Choices matter (Beta vs Logistic-Normal)
6. Multiple routes (constructive vs CLT)
7. Intellectual honesty (understand vs accept)

Applied to Generative Models:

- Incremental construction: simple → modern architectures
- Operations: transformations, mixtures, products, composition
- Frameworks: PGMs vs NNs (strengths, tradeoffs)
- Evolution through questions (why VAEs extend classics, why diffusion improves GANs)
- Markov property: from memoryless processes here to autoregressive models, diffusion processes
- Task-motivated organization (not just chronology)
- Cooperative dialectic (exploration, not passive reception)
- Honest about mysteries (practice ahead of theory)

Template established: Bernoulli → ecosystem of distributions. Simple generative models → modern AI architectures. Operations differ, choices multiply, but the *way of thinking* remains the same.

2 Foundational Reading

Books

- ”Probabilistic Machine Learning: An Introduction” Murphy [2022]
- ”Deep Learning Foundations and Concepts”, Bishop [2024]
- ”An Introduction to Probabilistic Graphical Models”, Jordan [2003]
- "Information Theory, Inference, and Learning Algorithms", MacKay [2003]
- ”Deep Learning”, Goodfellow et al. [2016]

Theses, Monographs, Tutorials

- “A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models”, Bilmes [1998]
- “Probabilistic Inference Using Markov Chain Monte Carlo Methods”, Neal [1993]
- “An Introduction to MCMC for Machine Learning”, Andrieu et al. [2003]
- “Variational Algorithms for Approximate Bayesian Inference”, Thesis, Beal [2003]
- “Graphical models for visual object recognition and tracking”, Thesis, Sudderth [2006]

Courses

- Stanford, Deep Generative Models CS236 (Stefano Ermon - Fall 2023)[1]
- UC Berkeley, CS294-158: Deep Unsupervised Learning (Pieter Abbeel, Jonathan Ho, Aravind Srinivas - Spring 2020) [2]/2024[3])
- CMU, 10708 - Probabilistic Graphical Models, (Andrej Risteski, Spring 2024) [4]

[1] https://deepgenerativemodels.github.io/
[2] https://sites.google.com/view/berkeley-cs294-158-sp20/home
[3] https://sites.google.com/view/berkeley-cs294-158-sp24/home
[4] https://andrejristeski.github.io/10708S24/

3 Foundations Through Simple Examples

3.1 Simple Generative Models

3.1.1 Example 1: Generating Heights of People

Consider the task of modeling the distribution of heights in a population.

Approach:

1. **Assume a distributional form**: Heights are approximately normally distributed

$$x \sim \mathcal{N}(\mu, \sigma^2) \tag{3}$$

2. **Estimate parameters from data**: Given observed population data $\mathcal{D} = \{x^{(1)}, \ldots, x^{(n)}\}$, estimate:

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x^{(i)} \tag{4}$$

$$\hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^{n} (x^{(i)} - \hat{\mu})^2 \tag{5}$$

3. **Generate synthetic data**: Sample new heights from the fitted distribution

$$x_{\text{new}}^{(i)} \sim \mathcal{N}(\hat{\mu}, \hat{\sigma}^2) \quad \text{for } i = 1, \ldots, m \tag{6}$$

This simple approach works well when:

- The true distribution is close to the assumed parametric form
- The number of parameters is small relative to data size
- Data points are independent and identically distributed (i.i.d.)

3.1.2 Example 2: Generating Sequences of Coin Tosses

For binary outcomes, we use a Bernoulli distribution.

Approach:

1. **Assume a distributional form**: Each toss follows a Bernoulli distribution

$$x \sim \text{Ber}(\rho) \tag{7}$$

where ρ is the probability of heads

2. **Estimate parameter from data**: Given observed toss data $\mathcal{D} = \{x^{(1)}, \ldots, x^{(n)}\}$ where $x^{(i)} \in \{0, 1\}$:

$$\hat{\rho} = \frac{1}{n} \sum_{i=1}^{n} x^{(i)} \tag{8}$$

3. **Generate synthetic tosses**:

$$x_{\text{new}}^{(i)} \sim \text{Ber}(\hat{\rho}) \quad \text{for } i = 1, \ldots, m \tag{9}$$

3.2 Beyond Independence: Non-IID Models

3.2.1 The IID Assumption and Its Limitations

The **independently and identically distributed (i.i.d.)** assumption states:

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i; \theta) \tag{10}$$

When IID Fails: Consider modeling daily temperatures:

- Temperature on day $t+1$ depends on temperatures of previous days
- X_{t+1} is **not independent** of $X_1, X_2, \ldots, X_t$
- X_{t+1} is **not identically distributed** to $X_1, X_2, \ldots, X_t$

3.2.2 Chain Rule of Probability

For any joint distribution, the chain rule gives exact factorization:

$$P(X_1, X_2, \ldots, X_n) = P(X_1)P(X_2|X_1)P(X_3|X_1, X_2) \cdots P(X_n|X_1, \ldots, X_{n-1}) \tag{11}$$

Computational Challenges:

- Requires **exponential space** to store all conditional probabilities
- Requires **exponential parameters** to model each conditional
- Requires **exponential time** to compute joint probabilities

3.2.3 Simplifying Assumptions via Conditional Independence

IID Modeling Assumption:

$$P(X_i|X_1, \ldots, X_{i-1}) = P(X_i) \quad \forall i \tag{12}$$

Complete independence: each variable independent of all others.

Markov Modeling Assumption:

$$P(X_i|X_1, \ldots, X_{i-1}) = P(X_i|X_{i-1}) \quad \forall i \tag{13}$$

First-order Markov: each variable depends only on immediate predecessor.

3.2.4 Factorization Through Conditional Independence

Conditional independence assumptions factorize the joint distribution into manageable terms:

IID Factorization:

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i; \theta) \tag{14}$$

Only $|\theta|$ parameters regardless of n.

Markov Factorization:

$$P(X_1, \ldots, X_n) = P(X_1; \theta_1) \prod_{i=2}^{n} P(X_i | X_{i-1}; \theta_i) \tag{15}$$

Linear growth in parameters: $|\theta_1| + (n-1)|\theta_2|$.

3.3 Generative Models for Classification

3.3.1 Generative vs Discriminative Tasks

Using Generative Models for Discriminative Tasks: A **generative model** learns $P(X, Y)$ (or $P(X|Y)$ and $P(Y)$). To perform classification (discriminative task), we must "invert" the model using Bayes' rule.

3.3.2 Generative Model for Two-Class Data

Consider binary classification with data X and label $Y \in \{0, 1\}$.

Joint Distribution:

$$P(X, Y) = P(Y; \theta) P(X|Y; \phi) \tag{16}$$

Algorithm 1 Generate Two-Class Data

1: **for** $i = 1$ to n **do**
2: $\quad Y_i \sim \text{Ber}(p)$
3: $\quad X_i \sim \mathcal{N}(\mu_{Y_i}, \sigma^2_{Y_i})$
4: **end for**

Generative Process: Different Gaussian distributions for each class.

3.3.3 Classification via Bayes' Rule

To classify a new data point x, compute posterior:

$$P(Y|X) = \frac{P(X, Y)}{P(X)} = \frac{P(X, Y)}{\sum_y P(Y = y, X)} \tag{17}$$

Expanding:

$$P(Y|X) = \frac{P(Y; \theta) P(X|Y; \phi)}{\sum_y P(Y = y; \theta) P(X|Y = y; \phi)} \tag{18}$$

Computational Challenge: The **denominator (normalization term)** requires summing/integrating over all possible values of Y and often becomes **intractable**

- High-dimensional X
- Complex dependencies in $P(X|Y)$
- Latent variables in the model

This intractability motivates approximate inference methods (variational inference, MCMC, etc.).

3.4 Latent Variables

3.4.1 Definition and Motivation

A **latent variable** (also called **hidden variable**) is an unobserved random variable that we introduce to help model observed data. We denote observed data as x and latent variables as z.

Why Introduce Unobserved Variables? Latent variables allow us to express complex distributions $p(x)$ through simpler joint distributions $p(x, z)$:

$$p(x) = \int p(x, z)\, dz = \int p(x|z)p(z)\, dz \tag{19}$$

The key insight: Even if $p(x)$ is complex, we can often find latent variables z such that $p(z)$ and $p(x|z)$ are both simple.

General Framework: A latent variable model specifies a joint distribution that factorizes as:

$$P(X, Z; \theta, \phi) = P(Z; \theta)P(X|Z; \phi) \tag{20}$$

The generative process:

Algorithm 2 Latent Variable Model Generation

1: **for** $i = 1$ to n **do**
2: $\quad Z_i \sim P(Z; \theta)$ // Sample latent variable
3: $\quad X_i \sim P(X|Z_i; \phi)$ // Generate observation given latent
4: **end for**

We now examine two important applications of latent variable models: mixture modeling and compression.

3.4.2 Generative Models for Mixtures

Motivating Example: Population Heights Consider modeling the heights of people in a population:

- **Observed data**: Heights X_i of individuals
- **Latent variable**: Ethnic group Z_i of each individual

Different ethnic groups may have different height distributions. The ethnic group information explains why certain heights are more common, but this information is not directly recorded.

Key Question: Can we unsupervised recover the ethnic groups from observed heights alone?

Gaussian Mixture Model (GMM): The GMM is a canonical example where the latent variable z is **discrete**, taking values in $\{1, 2, \ldots, K\}$. Each value represents one Gaussian component.

Algorithm 3 Gaussian Mixture Model

1: **for** $i = 1$ to n **do**
2: $\quad z_i \sim \text{Cat}(\boldsymbol{\alpha})$ // Sample component assignment
3: $\quad x_i \sim \mathcal{N}(\mu_{z_i}, \sigma^2_{z_i})$ // Sample from assigned Gaussian
4: **end for**

Generative Process:

Mathematical Formulation: Prior over components:

$$P(z_i = k) = \alpha_k \quad \text{where} \quad \sum_{k=1}^{K} \alpha_k = 1 \tag{21}$$

Conditional distribution:

$$P(x_i|z_i = k) = \mathcal{N}(x_i|\mu_k, \sigma_k^2) \tag{22}$$

Marginal distribution (mixture):

$$P(x_i) = \sum_{k=1}^{K} \alpha_k \mathcal{N}(x_i|\mu_k, \sigma_k^2) \tag{23}$$

Posterior Inference: To infer which cluster generated observation x_i, compute the posterior (called **responsibility**):

$$\gamma_{ik} = P(z_i = k|x_i) = \frac{\alpha_k \mathcal{N}(x_i|\mu_k, \sigma_k^2)}{\sum_{j=1}^{K} \alpha_j \mathcal{N}(x_i|\mu_j, \sigma_j^2)} \tag{24}$$

3.4.3 Generative Models for Compression

Motivating Example: Facial Images in the Same Population Returning to our population, now consider face photographs:

- **Observed data**: Face images $X_i \in \mathbb{R}^{256\times256\times3}$ (196,608 dimensions)
- **Latent variable**: Continuous facial features $Z_i \in \mathbb{R}^{512}$ such as skin tone gradients, facial structure parameters, expression coefficients

While ethnic group was a **discrete** latent variable for clustering heights, facial features form a **continuous** latent space. These features vary both within and across ethnic groups—two individuals from the same ethnic group can have different facial structures, expressions, and lighting.

Key Question: Can we learn a compressed representation that captures the essential features needed to reconstruct faces, while discarding redundant pixel-level information?

Compression via Generative Modeling: Consider data $X \in \mathbb{R}^d$ (e.g., $d = 196,608$ for face images) and latent code $Z \in \mathbb{R}^k$ where $k < d$ (e.g., $k = 512$, giving 400:1 compression). Here the latent variable z is **continuous** and provides a compressed representation of x.

Joint Distribution:

$$P(X, Z) = P(Z; \theta)P(X|Z; \phi) \tag{25}$$

where:

$$P(Z) = \mathcal{N}(0, I_k) \tag{26}$$

$$P(X|Z; \phi) = \mathcal{N}(\mu_\phi(Z), \sigma^2 I_d) \tag{27}$$

Here $\mu_\phi(Z)$ maps the latent code to the mean face image. In modern implementations, μ_ϕ is a neural network (the decoder).

Algorithm 4 Generate Data from Latent Code

for $i = 1$ to n **do**
  $Z_i \sim \mathcal{N}(0, I_k)$ // Sample latent code
  $X_i \sim \mathcal{N}(\mu_\phi(Z_i), \sigma^2 I_d)$ // Generate face from features
end for

Generative Process:

Encoding via Bayes' Rule: To compress a face image x into latent representation z, compute the posterior:

$$P(Z|X) = \frac{P(Z;\theta)P(X|Z;\phi)}{\int_z P(Z=z;\theta)P(X|Z=z;\phi)\,dz} \tag{28}$$

Compression Scheme:

- **Encode**: Given face image x, compute $P(z|x)$ to obtain compressed representation (512 facial feature values)
- **Decode**: Given z, sample $x \sim P(x|z)$ to reconstruct the face image
- **Lossy compression**: Some pixel-level detail is lost (196,608 $\rightarrow$ 512 dimensions)
- **Rate-distortion tradeoff**: How many features do we need to preserve recognizable faces?

3.4.4 The Computational Challenge

Both mixture models and compression models face the same fundamental challenge: computing the marginal likelihood $p(x)$ requires integrating (or summing) over all possible values of the latent variable z.

For Discrete Latent Variables (Mixtures):

$$P(x) = \sum_{k=1}^{K} P(x, z=k) = \sum_{k=1}^{K} P(z=k)P(x|z=k) \tag{29}$$

This becomes intractable when K is large or when computing posterior $P(z|x)$ for all z is expensive.

For Continuous Latent Variables (Compression):

$$P(x) = \int P(x,z)\,dz = \int P(z;\theta)P(x|z;\phi)\,dz \tag{30}$$

This integral is intractable for:

- High-dimensional latent spaces
- Complex dependencies in $P(X|Z)$
- Non-conjugate likelihood-prior pairs

This intractability is the **central challenge of latent variable models**. It motivates:

- **EM algorithm**: Iteratively optimizes a lower bound (for classical models like GMM)
- **Variational inference**: Approximates intractable posteriors with simpler distributions
- **Variational Autoencoders (VAEs)**: Use neural networks for amortized variational inference

Forward Connection: The compression interpretation provides intuition for VAEs: the encoder network $q_\phi(z|x)$ approximates the intractable true posterior $P(z|x)$, and the decoder network $p_\theta(x|z)$ is the generative model. The VAE framework makes this compression scheme tractable through amortized variational inference. The full framework is developed in Section 5.

4 Mathematical Formulations for Generative Modeling Tasks

4.1 Introduction

4.1.1 Key Insight

Machine learning $\neq$ just optimization. Multiple mathematical frameworks exist for each generative modeling task—optimization, Bayesian, sampling-based, and hybrid approaches. The choice matters: it affects computational complexity, approximation quality, uncertainty quantification, and scalability.

4.1.2 The Central Challenge

All four tasks face a common fundamental challenge: **high-dimensional complexity**. Real-world data (images, text, audio) lives in extremely high-dimensional spaces where:

- Naive approaches are computationally intractable
- Strong independence assumptions are unrealistic
- We must balance model expressiveness against computational feasibility

4.1.3 Fundamental Trade-offs

Machine learning's central tension is the **bias-variance tradeoff**: simple models underfit (high bias), complex models overfit (high variance). In generative modeling, this manifests in multiple ways:

Expressiveness vs Tractability

- **Simple models**: High bias, low variance, tractable (closed-form solutions)
- **Expressive models**: Low bias, high variance, intractable (require approximations)

Why inevitable: Marginalization over many variables with complex dependencies is exponentially expensive. Tractability requires either simplifying assumptions (increasing bias) or massive data (demanding computation).

Quality vs Diversity Probability mass is conserved (sums to 1), forcing allocation decisions:

- **Concentrate on common modes**: High quality, but mode collapse on rare cases
- **Spread across all modes**: Better coverage, but lower quality per mode

Why inevitable: Fitting what you observe well conflicts with maintaining coverage for what you haven't seen. Zero-sum allocation.

Sampling vs Density Architectures enable some operations while making others intractable:

- **Forward models** (e.g., GANs): Sampling trivial, density intractable
- **Density models** (e.g., autoregressive): Density direct, sampling sequential
- **Invertible models** (e.g., flows): Both efficient, but architectural constraints limit expressiveness

Why inevitable: Forward mappings ($z \to x$) and inverse problems ($x \to z$) have fundamentally different computational requirements.

4.1.4 This Section's Approach

We examine each of the four core tasks—sampling, scoring, inference, learning—through multiple mathematical lenses:

- Optimization (find single best point)
- Bayesian (maintain distributions, compute expectations)
- Sampling-based (Monte Carlo methods)
- Hybrid approaches

For each task: formulations → methods → when to use which.

4.2 Sampling (Generation)

4.2.1 The Task

Sampling is the task of drawing new data points from the learned distribution:

$$x \sim p(x|\theta) \tag{31}$$

Why Sampling is Primary: Generation is the fundamental act of bringing possibilities into existence. Before we can evaluate data (scoring), understand its structure (inference), or improve our model (learning), we must be able to create data points. In this sense, sampling is **ontologically prior**—it creates the space of actualized possibilities that other tasks operate on.

Why Sampling Matters:

- **Data augmentation**: Generate synthetic training examples
- **Simulation**: Create scenarios for planning and decision-making
- **Creative applications**: Generate images, text, music, designs
- **Privacy-preserving datasets**: Synthetic data that preserves statistical properties
- **Exploration**: Discover the range of possibilities the model has learned

The Computational Challenge: For simple models, sampling is straightforward. For complex models, especially those with:

- High-dimensional spaces
- Complex dependencies between variables
- Latent variables requiring inference
- Intractable normalizing constants

We need sophisticated sampling strategies that balance exactness, efficiency, and sample quality.

4.2.2 Direct Sampling

When the model structure permits, we can sample directly without approximation.

Ancestral Sampling For models with tractable conditional distributions:

$$p(x) = \int p(x|z)p(z)dz \tag{32}$$

Sample via:

1. Sample latent: $z \sim p(z)$
2. Sample data: $x \sim p(x|z)$

This extends naturally to deep hierarchies:

$$z_1 \sim p(z_1), \quad z_2 \sim p(z_2|z_1), \quad \ldots, \quad x \sim p(x|z_L) \tag{33}$$

Used in: VAEs, hierarchical models, Bayesian networks Koller and Friedman [2009]

Reparameterization Express a random variable as a deterministic transformation of simple noise:

$$z = g(\epsilon; \theta), \quad \epsilon \sim p(\epsilon) \tag{34}$$

Example (Gaussian): $z = \mu + \sigma\epsilon$ where $\epsilon \sim \mathcal{N}(0, 1)$
Why this matters:

- Enables low-variance gradient estimates Kingma and Welling [2014]
- Pathwise derivatives through random sampling
- Critical for training VAEs and normalizing flows Rezende and Mohamed [2015]

Advantages of Direct Sampling

- Exact samples from the model
- Fast generation (single forward pass)
- Deterministic computation graph (via reparameterization)
- No approximation error

Limitations

- Requires tractable conditional distributions
- Model architecture must support efficient sampling
- Not all distributions admit simple reparameterizations

4.2.3 MCMC Sampling

When direct sampling is intractable, Markov Chain Monte Carlo methods generate samples that asymptotically follow the target distribution.

Metropolis-Hastings General-purpose MCMC Metropolis et al. [1953], Hastings [1970]:

1. Propose: $x' \sim q(x'|x_t)$
2. Accept with probability: $\alpha = \min\left(1, \frac{p(x')q(x_t|x')}{p(x_t)q(x'|x_t)}\right)$
3. If accepted: $x_{t+1} = x'$, else: $x_{t+1} = x_t$

Gibbs Sampling When conditional distributions are tractable Geman and Geman [1984]:

$$x_i^{(t+1)} \sim p(x_i | x_1^{(t+1)}, \ldots, x_{i-1}^{(t+1)}, x_{i+1}^{(t)}, \ldots, x_n^{(t)}) \tag{35}$$

Cycles through variables, sampling each conditional on all others.

Hamiltonian Monte Carlo Gradient-based MCMC Duane et al. [1987], Neal [2011]:

- Introduces auxiliary momentum variables
- Simulates Hamiltonian dynamics using gradients
- Explores high-dimensional spaces efficiently
- Reduces random-walk behavior of simple MCMC

Langevin Dynamics Stochastic gradient Langevin dynamics Welling and Teh [2011]:

$$x_{t+1} = x_t + \frac{\epsilon}{2} \nabla_x \log p(x_t) + \sqrt{\epsilon} \eta_t, \quad \eta_t \sim \mathcal{N}(0, I) \tag{36}$$

Combines gradient ascent with noise injection. Converges to $p(x)$ as $\epsilon \to 0$ and iterations $\to \infty$.

MCMC Trade-offs :

Advantages:

- Asymptotically exact (given sufficient mixing)
- Flexible—works for any distribution we can evaluate
- No restricting architectural constraints
- Provides diverse samples after burn-in

Disadvantages:

- Requires many iterations to converge
- Mixing time can be slow, especially in high dimensions
- Must discard burn-in samples
- Difficult to parallelize across the chain
- Convergence diagnostics are non-trivial

4.2.4 Controlling Generation

Rather than purely sampling from $p(x|\theta)$, we often want to control properties of generated samples.

Temperature Scaling Control randomness via temperature parameter T:

$$p_T(x) \propto p(x)^{1/T} \tag{37}$$

- $T \to 0$: Deterministic, concentrates on modes
- $T = 1$: Original distribution
- $T > 1$: More uniform, increased diversity

Sampling Strategies for Sequences :
For autoregressive models generating sequences $x_1, \ldots, x_T$:

- **Greedy decoding**: $x_t = \arg\max p(x_t|x_{<t})$
  - Deterministic, locally optimal
  - Can be repetitive or get stuck
- **Ancestral sampling**: $x_t \sim p(x_t|x_{<t})$
  - Stochastic, diverse
  - Can produce low-quality outliers
- **Top-k sampling** Fan et al. [2018]: Sample from top k most probable tokens
  - Balances quality and diversity
  - Fixed cutoff may be too rigid
- **Nucleus (top-p) sampling** Holtzman et al. [2020]: Sample from smallest set with cumulative probability $\geq p$
  - Adaptive cutoff based on distribution shape
  - Widely used in modern language models
- **Beam search**: Maintain k best partial sequences
  - Approximates most likely complete sequence
  - Can lack diversity across beams

Quality vs. Diversity Trade-off Deterministic methods (greedy, low temperature) produce consistent, high-quality samples but lack diversity and can be repetitive. Stochastic methods (ancestral, high temperature) produce diverse samples but with variable quality. The choice depends on the application: creativity benefits from diversity, while safety-critical applications may prefer deterministic high-quality outputs.

4.2.5 When Each Applies

- **Direct sampling**: When model has tractable conditional distributions (VAEs, autoregressive models, normalizing flows)
- **MCMC**: When direct sampling intractable but can evaluate $p(x)$ (energy-based models, complex posteriors)
- **Temperature/sampling strategies**: When fine-tuning the quality-diversity trade-off for specific applications

4.3 Scoring (Density Estimation)

4.3.1 The Task

Scoring is the task of evaluating the probability (or probability density) of data under the learned model:

$$\text{Compute: } p(x|\theta) \text{ for given data point } x \tag{38}$$

Why Scoring Matters:

- **Model evaluation**: Compare models via likelihood on held-out data
- **Anomaly detection**: Low probability indicates unusual/anomalous data
- **Likelihood-based learning**: Many learning algorithms require computing $p(x|\theta)$
- **Data compression**: Optimal encoding length determined by $-\log p(x)$

The Computational Challenge: For models *without* latent variables, scoring is straightforward—just evaluate the density function.

For models *with* latent variables, scoring requires computing the marginal likelihood:

$$p(x|\theta) = \int p(x, z|\theta)\, dz = \int p(x|z, \theta)p(z|\theta)\, dz \tag{39}$$

This integral is typically **intractable** due to:

- High-dimensional latent spaces
- Complex dependencies in $p(x|z, \theta)$
- No closed-form solution for the integral

This intractability is the central challenge of scoring for latent variable models, and different approaches address it through direct computation (when possible), optimization-based bounds, or sampling-based estimation.

4.3.2 Direct Computation

When the model structure permits, we can compute $p(x|\theta)$ directly:

$$p(x|\theta) = \frac{1}{Z(\theta)}\tilde{p}(x|\theta) \tag{40}$$

where $\tilde{p}(x|\theta)$ is the unnormalized density and $Z(\theta) = \int \tilde{p}(x|\theta)dx$ is the normalizing constant (partition function).

When Direct Computation is Tractable Direct computation is feasible for:

- **Simple parametric models**: Gaussian distributions, exponential families with known sufficient statistics Wainwright and Jordan [2008]
- **Autoregressive models**: Factorized densities $p(x) = \prod_t p(x_t|x_{<t})$ Bengio et al. [2000]
- **Normalizing flows**: Invertible transformations with tractable Jacobians Rezende and Mohamed [2015], Dinh et al. [2017b]
- **Small discrete spaces**: Enumerable support where we can sum over all values

Advantages

- Exact likelihood values
- Principled model comparison via likelihood ratios
- Unbiased gradient estimates for learning
- No approximation error

Limitations

- Restricted to specific model families
- Architectural constraints needed to maintain tractability
- Computational cost grows with model complexity

4.3.3 Optimization-Based Formulations

When exact computation is intractable, we can derive bounds or reformulate as optimization.

Evidence Lower Bound (ELBO) For latent variable models where $p(x|\theta) = \int p(x, z|\theta)dz$ is intractable, the ELBO provides a lower bound Jordan et al. [1999]:

$$\log p(x|\theta) \geq \mathbb{E}_{q(z|x)}[\log p(x, z|\theta) - \log q(z|x)] = \mathcal{L}(x; \theta, q) \tag{41}$$

The gap between the bound and true log-likelihood is:

$$\log p(x|\theta) - \mathcal{L}(x; \theta, q) = D_{\mathrm{KL}}(q(z|x)\|p(z|x, \theta)) \tag{42}$$

Optimizing q to maximize the ELBO tightens this bound, providing increasingly accurate approximations to the true log-likelihood.

Implicit Models Generative Adversarial Networks Goodfellow et al. [2014] define no explicit density function. Scoring in such models requires:

- Training a separate density estimator on generated samples
- Using the discriminator as a proxy score
- Likelihood-free inference methods

4.3.4 Sampling-Based Estimation

When neither direct computation nor optimization-based bounds are suitable, we resort to Monte Carlo estimation.

Importance Sampling For latent variable models:

$$p(x|\theta) = \int p(x, z|\theta)dz = \mathbb{E}_{q(z)}\left[\frac{p(x, z|\theta)}{q(z)}\right] \tag{43}$$

Estimate via samples from proposal q:

$$\hat{p}(x|\theta) = \frac{1}{K}\sum_{k=1}^{K}\frac{p(x, z_k|\theta)}{q(z_k)}, \quad z_k \sim q(z) \tag{44}$$

Key challenge: Requires a good proposal distribution $q(z)$ to avoid high-variance estimates. Poor proposals lead to a few samples dominating the sum.

Application: Importance Weighted Autoencoder Burda et al. [2016] uses multiple importance samples for tighter likelihood bounds.

Simple Monte Carlo When we can sample from the prior $p(z|\theta)$:

$$p(x|\theta) = \int p(x|z,\theta)p(z|\theta)dz \approx \frac{1}{K}\sum_{k=1}^{K} p(x|z_k,\theta), \quad z_k \sim p(z|\theta) \tag{45}$$

This is unbiased but can have high variance, requiring many samples for accurate estimates.

4.3.5 When Each Applies

- **Direct computation**: When model structure permits (flows, autoregressive, simple parametric models)
- **ELBO**: Latent variable models where we can optimize a variational distribution (VAEs, approximate inference)
- **Importance sampling**: When a good proposal distribution is available
- **Simple Monte Carlo**: When we can sample from the prior but the likelihood is tractable
- **No explicit scoring**: For implicit models (GANs) where generation is primary goal

4.4 Inference (Posterior Computation)

4.4.1 The Task

Inference is the task of computing the posterior distribution over latent variables given observed data:

$$\text{Compute: } p(z|x,\theta) \text{ for given data point } x \tag{46}$$

Using Bayes' rule:

$$p(z|x,\theta) = \frac{p(x,z|\theta)}{p(x|\theta)} = \frac{p(x|z,\theta)p(z|\theta)}{\int p(x,z|\theta)dz} \tag{47}$$

Why Inference Matters:

- **Representation learning**: Discover meaningful latent structure in data
- **Encoding**: Compress high-dimensional data to low-dimensional codes
- **Missing data imputation**: Fill in unobserved values
- **Denoising**: Recover clean signals from corrupted observations
- **Clustering**: Discover groupings in data
- **Essential for learning**: When latent variables present, learning requires inference as subtask (EM algorithm)

The Computational Challenge - Shared with Scoring: Notice that computing the posterior requires the **same intractable marginal** as scoring:

$$p(x|\theta) = \int p(x,z|\theta)dz \tag{48}$$

This appears in:

- **Scoring task**: Compute $p(x|\theta)$ directly

- **Inference task**: Compute $p(z|x, \theta) = p(x, z|\theta)/p(x|\theta)$ (denominator!)

Both tasks face the same fundamental obstacle: marginalizing over latent variables is intractable for complex models. This shared challenge is why variational methods can solve both tasks simultaneously—optimizing the ELBO provides:

- A lower bound on $\log p(x|\theta)$ (for scoring)
- An approximation $q(z|x)$ to $p(z|x, \theta)$ (for inference)

Different approaches address this intractability through exact computation (when possible), point estimates (MAP), approximate distributions (variational inference), or sampling (MCMC).

4.4.2 Full Posterior: The Gold Standard

The complete solution is computing the entire posterior distribution:

$$p(z|x, \theta) = \frac{p(x, z|\theta)}{\int p(x, z|\theta) dz} \tag{49}$$

Benefits

- Complete uncertainty quantification
- Optimal for decision-making via posterior expectations
- No information loss
- Enables computing any statistic: $\mathbb{E}_{p(z|x)}[f(z)]$

When Tractable Exact posterior computation is possible for:

- **Conjugate models**: Prior and posterior in same family Murphy [2012]
- **Linear-Gaussian models**: Kalman filtering Kalman [1960]
- **Discrete small state spaces**: Enumeration feasible
- **Tree-structured graphical models**: Belief propagation Pearl [1988]

For most interesting models with high-dimensional latent spaces and complex dependencies, exact computation is intractable. We need approximations.

4.4.3 MAP Inference: Point Estimates

Maximum a posteriori (MAP) inference finds the mode:

$$z^* = \arg\max_z p(z|x, \theta) = \arg\max_z [\log p(x|z, \theta) + \log p(z|\theta)] \tag{50}$$

Interpretation MAP provides a single point estimate—the most probable latent configuration. Equivalent to:

- Penalized maximum likelihood (prior = regularization)
- Limit of Bayesian inference as posterior sharpens

Trade-offs :

Advantages:

- Computationally efficient (single optimization)
- Deterministic output
- Standard optimization tools apply

Disadvantages:

- No uncertainty quantification
- May miss important probability mass
- Overconfident in high dimensions MacKay [2003]

When appropriate: Single best answer needed, computational budget limited, posterior strongly peaked, uncertainty unnecessary.

4.4.4 Variational Inference: Optimization-Based Approximation

Variational inference converts the intractable integration problem into a tractable optimization problem Jordan et al. [1999], Blei et al. [2017].

Core Idea Instead of computing $p(z|x, \theta)$ directly, introduce a tractable variational distribution $q_\phi(z|x)$ from family $\mathcal{Q}$ and find the best approximation:

$$q^*(z|x) = \arg\min_{q \in \mathcal{Q}} D_{\mathrm{KL}}(q(z|x) \| p(z|x, \theta)) \tag{51}$$

Equivalently, maximize the Evidence Lower Bound (ELBO):

$$\mathcal{L}(x; \theta, \phi) = \mathbb{E}_{q_\phi(z|x)}[\log p(x, z|\theta) - \log q_\phi(z|x)] \tag{52}$$

The ELBO decomposes as:

$$\log p(x|\theta) = \mathcal{L}(x; \theta, \phi) + D_{\mathrm{KL}}(q_\phi(z|x) \| p(z|x, \theta)) \tag{53}$$

Maximizing ELBO w.r.t. ϕ minimizes the KL divergence, making $q_\phi(z|x)$ a better approximation to the true posterior.

Classical Mean Field Approximation Traditional variational inference assumed complete factorization:

$$q_\phi(z|x) = \prod_{i=1}^{d} q_{\phi_i}(z_i|x) \tag{54}$$

Additional restrictions for tractability:

- Exponential family distributions with analytical expectations
- Coordinate ascent optimization
- Separate parameters per data point

Modern Amortized Inference :
Modern approaches (e.g., VAEs) use neural networks:

- **Encoder network** $q_\phi(z|x)$: Maps data to approximate posterior
- **Amortization**: Single network handles all data points
- **Reparameterization trick**: Low-variance gradient estimates Kingma and Welling [2014]
- **End-to-end learning**: Joint optimization of inference and generative model

Trade-offs :
Advantages:

- Fast (optimization-based, not sampling)
- Scalable to large datasets (minibatch SGD)
- Deterministic
- Provides both posterior approximation AND likelihood bound

Disadvantages:

- Biased (underestimates posterior variance)
- Restricted to chosen family $\mathcal{Q}$
- May miss modes
- Amortization gap in amortized inference Cremer et al. [2018]

4.4.5 Markov Chain Monte Carlo: Sampling-Based Approximation

MCMC methods generate samples that asymptotically follow the posterior Metropolis et al. [1953], Hastings [1970]:

$$z_1, z_2, \ldots, z_K \sim p(z|x, \theta) \tag{55}$$

Common Methods

- **Metropolis-Hastings**: General-purpose with proposal and accept/reject
- **Gibbs sampling**: Sample each variable conditional on others Geman and Geman [1984]
- **Hamiltonian Monte Carlo**: Uses gradients for efficient exploration Duane et al. [1987], Neal [2011]
- **Langevin dynamics**: Gradient-based with noise injection Welling and Teh [2011]

Trade-offs :
Advantages:

- Asymptotically exact
- Flexible—works for any posterior we can evaluate
- No restricting approximation family
- Provides samples for expectation estimation

Disadvantages:

- Slow (many iterations needed)
- Mixing time uncertain
- Requires tuning (step sizes, proposals)
- Difficult to parallelize
- Must discard burn-in samples

4.4.6 When Each Applies

- **Full posterior**: Small models, conjugate structure, exact solution needed
- **MAP**: Speed critical, uncertainty unimportant, single best answer sufficient
- **Variational inference**: Large-scale problems, fast inference needed, acceptable bias
- **MCMC**: Accuracy critical, time available, asymptotic exactness needed
- **Hybrid approaches**: VI for initialization, MCMC for refinement Salimans et al. [2015]

The choice depends on:

- Model complexity and tractability
- Dataset size and computational budget
- Uncertainty requirements
- Speed vs. accuracy trade-offs

4.5 Learning (Parameter Estimation)

4.5.1 The Task

Learning is the task of estimating model parameters θ from observed data:

$$\text{Given data } \mathcal{D} = \{x_1, \ldots, x_n\}, \text{ find } \theta^* \tag{56}$$

The fundamental goal is to make our model distribution $p_\theta(x)$ match the true data distribution $p_{\text{data}}(x)$.

Why Learning Matters:

- **Enables all other tasks**: Without learned parameters, we cannot sample, score, or infer
- **Captures data patterns**: Extracts structure from observations to generalize
- **Adapts models**: Tailors general frameworks to specific domains
- **Foundation for transfer**: Pre-trained models fine-tuned for new tasks

The Computational Challenge - Builds on Scoring and Inference :

Learning depends critically on the previous tasks:

Connection to Scoring: Most learning objectives require evaluating $p(x|\theta)$:

- **Maximum Likelihood**: Maximize $\sum_{i=1}^{n} \log p(x_i|\theta)$
- **Bayesian learning**: Compute posterior $p(\theta|\mathcal{D}) \propto p(\mathcal{D}|\theta)p(\theta)$

For models with latent variables, this requires the intractable marginal:

$$p(x|\theta) = \int p(x, z|\theta)dz \tag{57}$$

Connection to Inference: When latent variables are present, learning and inference are intertwined:

- **EM algorithm**: Alternates between inference (E-step) and learning (M-step)
- **Variational learning**: Jointly optimizes θ and inference network ϕ
- Cannot learn without inferring latent structure

The dependency chain is complete: **Sampling enables Scoring, Scoring enables Inference, and Inference enables Learning**.

Different approaches address the learning problem through various distance metrics and optimization strategies, each with distinct properties and trade-offs.

4.5.2 The Goal: Measuring Distributional Distance

Embracing Variance: The Fundamental Challenge Variance is the wellspring of creation—we must embrace the full distribution p_{data}. But how exactly? We face constraints: limited model capacity, finite computation, approximate inference. Different distance metrics embody different philosophies about embracing variance under these constraints.

The core tension: When approximating p_{data} with limited-capacity p_θ:

- **Mode-covering**: Spread capacity across all modes? (acknowledge diversity, but outputs averaged/blurry)
- **Mode-seeking**: Focus capacity on few modes? (sharp outputs, but miss diversity)

There is no single perfect way—just different tradeoffs for different needs. This is why multiple distance metrics exist.

Kullback-Leibler (KL) Divergence The most common choice, arising from maximum likelihood estimation:

$$D_{\text{KL}}(p_{\text{data}} \| p_\theta) = \mathbb{E}_{x \sim p_{\text{data}}}[\log p_{\text{data}}(x) - \log p_\theta(x)] \tag{58}$$

Properties:

- Forward KL (as above) is **mode-covering**: p_θ tries to explain all variation in data
- Explains why ML often produces blurry/averaged outputs in generation
- Reverse KL $D_{\text{KL}}(p_\theta \| p_{\text{data}})$ is **mode-seeking**: p_θ focuses on peak regions
- Asymmetric: $D_{\text{KL}}(p\|q) \neq D_{\text{KL}}(q\|p)$
- Tractable gradients via log-likelihood

Alternative Distances - Different Tradeoffs :

Wasserstein Distance (Earth Mover's):

- Geometric interpretation: minimum cost to transport mass from p to q
- Captures closeness in **data space** (Euclidean position), not distribution space
- Works even when supports don't overlap (KL would be infinite)
- Used in Wasserstein GANs—but GANs escaped mode-covering only to face *mode collapse*!
- Historical debt: Monge (1781), Kantorovich (1940s)

f-Divergences (General Family):

- Unifying framework: $D_f(p\|q) = \mathbb{E}_q[f(dp/dq)]$
- Different f functions $\rightarrow$ different mode-covering/seeking behaviors
- Examples: KL, reverse KL, Jensen-Shannon, χ^2, total variation

Maximum Mean Discrepancy (MMD):

- Kernel-based distance, rooted in two-sample testing
- MMD-GANs circle back to principled statistical foundations
- Historical debt: Kolmogorov-Smirnov (1933/39) $\rightarrow$ MMD (Gretton et al., 2012)

Why We Focus on KL :

Despite its limitations, KL dominates because:

- Direct equivalence to maximum likelihood
- Computationally tractable gradients
- Information-theoretic interpretation (coding theory, compression)

But awareness of alternatives is essential—they represent different answers to the fundamental question: *How do we embrace variance given our constraints?*

4.5.3 Maximum Likelihood Estimation

KL Minimization Equals MLE :

Minimizing KL divergence is equivalent to maximizing log-likelihood:

$$\arg\min_\theta D_{\mathrm{KL}}(p_{\mathrm{data}}\|p_\theta) = \arg\min_\theta \mathbb{E}_{x\sim p_{\mathrm{data}}}[\log p_{\mathrm{data}}(x) - \log p_\theta(x)] \tag{59}$$

$$= \arg\max_\theta \mathbb{E}_{x\sim p_{\mathrm{data}}}[\log p_\theta(x)] \tag{60}$$

$$= \arg\max_\theta \sum_{i=1}^{n} \log p_\theta(x_i) \tag{61}$$

The first term $\mathbb{E}[\log p_{\mathrm{data}}(x)]$ is constant w.r.t. θ.

Properties :
Under regularity conditions, MLE is Lehmann and Casella [1998]:

- **Consistent**: Converges to true parameters as $n \to \infty$
- **Asymptotically efficient**: Achieves Cramér-Rao lower bound
- **Asymptotically normal**: $\sqrt{n}(\hat{\theta} - \theta_0) \to \mathcal{N}(0, I^{-1})$ where I is Fisher information
- **Invariant**: Under reparameterization

Frequentist Perspective :
MLE treats parameters as fixed unknown constants. Uncertainty characterized through:

- Confidence intervals from Fisher information
- Standard errors
- Bootstrap resampling

4.5.4 Bayesian Learning

Bayesian learning treats parameters as random variables with prior $p(\theta)$ and computes the posterior:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{\int p(\mathcal{D}|\theta)p(\theta)d\theta} \tag{62}$$

Full Bayesian Analysis :
Provides:

- **Posterior distribution**: Complete uncertainty over parameters
- **Posterior predictive**: $p(x'|\mathcal{D}) = \int p(x'|\theta)p(\theta|\mathcal{D})d\theta$
- **Credible intervals**: Bayesian uncertainty quantification
- **Model comparison**: Marginal likelihoods $p(\mathcal{D})$

Trade-offs :
Advantages:

- Principled uncertainty quantification
- Natural handling of small data
- Coherent sequential updating
- Automatic regularization through prior
- Optimal decision-making framework

Disadvantages:

- Computational cost (posterior often intractable)
- Prior specification sensitivity
- Requires approximations in practice (VI, MCMC)
- More complex than point estimates

4.5.5 MAP Estimation

MAP estimation finds the mode of the posterior:

$$\theta^* = \arg\max_\theta p(\theta|\mathcal{D}) = \arg\max_\theta [\log p(\mathcal{D}|\theta) + \log p(\theta)] \tag{63}$$

Connection to Regularization :
MAP with Gaussian prior $p(\theta) = \mathcal{N}(0, \lambda^{-1}I)$ is equivalent to:

$$\theta^* = \arg\max_\theta \log p(\mathcal{D}|\theta) - \frac{\lambda}{2}\|\theta\|^2 \tag{64}$$

This is penalized maximum likelihood (L2 regularization).
Different priors induce different regularizers:

- Gaussian prior: L2 regularization (ridge)
- Laplace prior: L1 regularization (lasso)
- Student-t prior: Robust to outliers

Relationship to MLE and Full Bayes :
MAP is:

- MLE with uniform prior: $\theta_{\text{MAP}} = \theta_{\text{MLE}}$
- Mode of Bayesian posterior (single point estimate)
- Approximates posterior mean when posterior is symmetric and unimodal

4.5.6 Other Formulations

Score Matching :
Match score functions (gradients of log-density) Hyvärinen [2005]:

$$\theta^* = \arg\min_\theta \mathbb{E}_{p_{\text{data}}} \left[\|\nabla_x \log p_\theta(x) - \nabla_x \log p_{\text{data}}(x)\|^2 \right] \tag{65}$$

Avoids computing partition function. Extended to denoising score matching Vincent [2011], which forms the basis of diffusion models Song et al. [2021b].

Adversarial Learning :
GANs Goodfellow et al. [2014] learn via discriminator objective:

$$\min_\theta \max_\phi \mathbb{E}_{x\sim p_{\text{data}}}[\log D_\phi(x)] + \mathbb{E}_{x\sim p_\theta}[\log(1 - D_\phi(x))] \tag{66}$$

Discriminator D_ϕ distinguishes real from generated samples.

4.5.7 When to Use Each

- **MLE**: Large data, asymptotic regime, unbiased estimator needed
- **Full Bayesian**: Small data, uncertainty critical, hierarchical models
- **MAP**: Want regularization, prior available, point estimate sufficient
- **Score matching**: Partition function intractable, gradient-based models (diffusion)
- **Adversarial**: Implicit models, generation quality paramount (GANs)

4.5.8 Types of Supervision

When learning generative models, the training signal (supervision) can come from different sources:

- **Unlabeled data**: Learn patterns from raw observations
- **Data structure**: Extract supervision from inherent structure (temporal, spatial)
- **Explicit labels**: Use human-provided annotations
- **Human preferences**: Learn from comparisons and rankings

The choice of supervision type depends on:

- What data is available
- Task requirements (control vs diversity)
- Resource constraints (labeling cost, compute)

The Supervision Spectrum :

Unsupervised Learning: Learn $P(x)$ from unlabeled data.

- **Examples**: VAEs, GANs, diffusion models on raw images/text
- **Advantages**: Abundant unlabeled data, no labeling cost
- **Limitations**: No direct control over generation, hard to steer outputs

Self-Supervised Learning: Create supervision from data structure itself.

- **Key insight**: Data provides its own supervision signal
- **Examples**:
  - Next-token prediction (language models)
  - Masked prediction (BERT-style)
  - Contrastive learning
  - Denoising (diffusion models)
- **Advantages**: Leverages structure, scalable, no external labels

Supervised Learning: Learn $P(x|y)$ with explicit labels y.

- **Examples**: Class-conditional generation, text-to-image (Stable Diffusion)
- **Advantages**: Controllable generation, task-specific
- **Limitations**: Expensive labels, limited labeled data

Learning from Preferences and Comparisons :

Motivation:

- Comparisons easier than absolute judgments ("A better than B" vs "A is 7/10")
- Subjective qualities (helpfulness, safety, aesthetics) hard to score directly
- Need to align models with human values/preferences

Preference-Based Learning:

- Collect pairwise comparisons: which output better?
- Model preferences via Bradley-Terry: $P(x_1 \succ x_2) = \frac{\exp(r(x_1))}{\exp(r(x_1))+\exp(r(x_2))}$
- Learn reward/scoring function from comparisons

Reinforcement Learning from Human Feedback (RLHF):
Three-stage process Ouyang et al. [2022]:

1. **Supervised pre-training**: Learn base model on demonstrations
2. **Reward modeling**: Train reward model on human preferences
3. **RL fine-tuning**: Optimize policy against learned reward (PPO)

Canonical example: InstructGPT/ChatGPT alignment
Why it matters: Aligns model behavior with human values beyond raw likelihood
Alternatives:

- **Direct Preference Optimization (DPO)** Rafailov et al. [2023]: Skip reward model, optimize preferences directly
- **Constitutional AI** Bai et al. [2022]: Self-critique and refinement via principles

Choosing Supervision Types :

Decision factors:

- **Data availability**: Labeled vs unlabeled, amount, quality
- **Task requirements**: Unconditional vs conditional, controllability needs
- **Resource constraints**: Compute budget, access to human feedback
- **Desired properties**: Diversity vs control, safety vs capability

Common patterns:

- **Foundation models**: Unsupervised/self-supervised pre-training at scale
- **Task-specific models**: Supervised fine-tuning for particular applications
- **Human alignment**: Preference-based refinement (RLHF, DPO)

Tradeoffs:

- Data efficiency $\leftrightarrow$ Label cost
- Generation quality $\leftrightarrow$ Controllability
- Computational cost $\leftrightarrow$ Alignment quality
- Model capability $\leftrightarrow$ Safety guarantees

Summary

- Multiple supervision types address different learning scenarios
- Choice depends on available data, task objectives, resources
- **Modern approach**: Combine supervision types sequentially
  - Pre-train (unsupervised/self-supervised)
  - Fine-tune (supervised for task)
  - Align (preferences for values)
- Each stage leverages different training signals optimally

4.6 Summary

This section has demonstrated that the core tasks of generative modeling—scoring, inference, learning, and sampling—admit multiple mathematical formulations, each with distinct properties and trade-offs.

Key Insights

1. **Not all machine learning is optimization**: Bayesian methods focus on computing distributions and expectations, not just finding modes
2. **Multiple valid approaches**: The same task can be formulated as optimization, Bayesian inference, expectation computation, or other frameworks
3. **Trade-offs are fundamental**: No universally best approach—choices depend on problem structure, computational resources, and task requirements
4. **Frameworks are connected**: MAP as mode of posterior, variational inference as optimization for integration, MCMC vs. optimization as exploration vs. exploitation
5. **Context determines choice**: Consider tractability, scalability, uncertainty requirements, and approximation quality when selecting a formulation

Practical Guidelines

- Start simple: Try optimization first for tractability
- Add complexity as needed: Incorporate uncertainty if critical
- Understand approximations: Know what you're sacrificing
- Use diagnostic tools: Validate approximation quality
- Match method to scale: Different approaches for different model sizes

Looking Forward: Modern generative AI increasingly relies on hybrid approaches:

- Optimization for learning (scalability)
- Variational inference for latent variables (tractability)
- Sampling for generation (quality and diversity)
- Bayesian methods for small-data or high-stakes domains (uncertainty)

5 Meta-Task: Modeling Complex Distributions

5.1 Introduction

Modeling is the foundational task:

- Choose what distribution to represent
- Construct it from operations and frameworks
- Must handle diverse data types:
  - Scalars, vectors (simple measurements)
  - Sequences (text, time series, audio) - 1D structure
  - Images (grids) - 2D structure
  - Video (spatiotemporal) - 3D structure
  - Graphs, point clouds, molecules - irregular structure
  - Mixed types
- How to best model each type emerges through discussion

This section covers:

- Operations for building complexity
- Two complementary frameworks (PGMs, NNs)
- How they combine to enable modern generative modeling

5.2 Operations for Creating Complexity

5.2.1 Transformation of Random Variables

Key Idea: Apply deterministic transformations to simple distributions to create complex ones.

Change of Variables Formula: Scalar:

$$p_Y(y) = p_X(g(y)) \left| \frac{d}{dy} g(y) \right| \tag{67}$$

Multivariate:

$$p_{\mathbf{Y}}(\mathbf{y}) = p_{\mathbf{X}}(g(\mathbf{y})) \left| \det J_g(\mathbf{y}) \right| \tag{68}$$

where J_g is the Jacobian matrix, $g = f^{-1}$.

What the Jacobian Represents: The Jacobian matrix J_g has elements:

$$[J_g]_{ij} = \frac{\partial g_i}{\partial y_j} = \frac{\partial x_i}{\partial y_j} \tag{69}$$

This is not just mathematical notation—it captures the **interactions between dimensions**:

- $J_g[i, j]$ measures: How much does changing input dimension j (of $\mathbf{y}$) affect output dimension i (of $\mathbf{x}$)?
- Full Jacobian (all elements nonzero): Transformation couples all dimensions
- Sparse/structured Jacobian: Limited cross-dimensional interactions

Computational Implications: Computing $|\det J_g|$ requires $O(n^3)$ operations for $n \times n$ matrix:

- For high-dimensional data (images: $n \sim 200{,}000$), this is completely intractable
- This computational barrier constrains which transformations are feasible
- **Key insight**: Modeling all interactions is prohibitively expensive

The Design Trade-off: This reveals a fundamental choice in generative modeling:

- **Rich interactions**: Model complex cross-dimensional dependencies, pay $O(n^3)$ cost
- **Structured interactions**: Assume limited dependencies (sparsity, ordering), achieve $O(n)$ tractability

This is the structure-capacity trade-off through the lens of transformations. We'll see how normalizing flows navigate this trade-off through architectural design in Section 10.

Effect on Variance and Complexity: Transformations *reshape* variance:

- Don't add/remove modes
- Nonlinear transformations create dependencies
- Create complex geometric structures (manifolds)
- Complexity from warping space

Computational Cost: $O(n^3)$ determinant computation prohibitive for high dimensions.

Normalizing Flows solution: Carefully designed transformations with tractable Jacobians (triangular, coupling layers, autoregressive).

5.2.2 Mixtures of Distributions (Probabilistic Sums)

Key Idea: Convex combination creates complex distribution. Probabilistic "OR" - sample from one component.

Definition:

$$p(x) = \sum_{i=1}^{k} \pi_i p_i(x) \quad \text{where} \quad \sum_{i=1}^{k} \pi_i = 1 \tag{70}$$

Latent Variable Interpretation:

$$P(Z = i) = \pi_i \tag{71}$$

$$p(x) = \sum_{i=1}^{k} P(Z = i)\, p(x|Z = i) \tag{72}$$

Effect on Variance: Mixtures *increase* variance:

- Add modes (multimodality)
- $\text{Var}[X] = \mathbb{E}_Z[\text{Var}[X|Z]] + \text{Var}_Z[\mathbb{E}[X|Z]]$
- Mixture variance $\geq$ average component variance
- Captures diversity, heterogeneous populations

Example: Gaussian Mixture Model (GMM)

$$p(x) = \sum_{i=1}^{k} \pi_i \mathcal{N}(x|\mu_i, \Sigma_i) \tag{73}$$

5.2.3 Products: Combining Constraints

The Operation: Multiply non-negative functions:

$$f(x) = \prod_{i=1}^{K} g_i(x_i) \tag{74}$$

where each $g_i \geq 0$. Then normalize (if needed):

$$p(x) = \frac{1}{Z} f(x), \quad Z = \int f(x)dx \tag{75}$$

The Intuition: Products combine multiple sources of information/constraints:

- Each g_i encodes preferences or compatibility for part of x
- Multiplication means: satisfy ALL constraints simultaneously
- Higher product value $\rightarrow$ configuration more compatible with all factors

When normalized: If the g_i are already probability distributions over independent variables, the product is automatically normalized (no Z needed). Otherwise, we compute the partition function Z to normalize.

Forward: We'll see how graphical models (Section X) use products to factorize complex distributions, with different normalization requirements for directed vs undirected models.

Effect on Variance: Products *decrease* variance:

- Sharpen distribution (more peaked)
- Combine constraints/information
- Each expert restricts possibilities

Challenges:

- Computing Z often intractable
- Connection to energy-based models
- Approximate methods needed

5.2.4 Composition of Operations

Combining operations creates even more complexity:

- Sequential application
- Neural networks as composed transformations
- Hierarchical latent variable models as composed mixtures
- Modern models use multiple operations

5.3 Two Frameworks: PGMs and Neural Networks

5.3.1 Probabilistic Graphical Models: The Natural Framework

Three Equivalent Views: 1. Probabilistic Program View:
Generative process as algorithm:

Algorithm 5 GenerateTosses(α, β, γ)

for $i = 1$ to k **do**
  $\theta_i \sim \text{Beta}(\alpha, \beta)$
end for
$\pi \sim \text{Dir}(\gamma)$
for $j = 1$ to N **do**
  $z_j \sim \text{Cat}(\pi)$
  $x_j \sim \text{Ber}(\theta_{z_j})$
end for

2. Factorization View:

$$P(x, z, \theta, \pi) = P(\pi) \prod_i P(\theta_i) \prod_j P(z_j|\pi) P(x_j|\theta_{z_j}) \tag{76}$$

Key: Factorization encodes conditional independence, not just chain rule. Represents domain knowledge and inductive bias.

3. Graphical View:

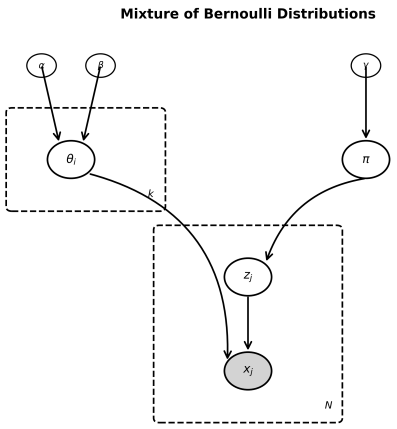

Figure 1: Plate diagram for mixture of Bernoulli distributions. Hyperparameters α, β, and γ (small circles) parameterize the priors. The outer plate (k) represents the k mixture components with Bernoulli parameters $\theta_i \sim \text{Beta}(\alpha, \beta)$. The mixing distribution $\pi \sim \text{Dir}(\gamma)$ governs component selection. The inner plate (N) represents the N observations: for each observation, the latent variable $z_j \sim \text{Cat}(\pi)$ selects a component, and the observed outcome $x_j \sim \text{Ber}(\theta_{z_j})$ (shaded) is drawn from that component's Bernoulli distribution.

- Nodes = random variables
- Edges = direct dependencies
- Shading = observed/unobserved
- Plates = repeated structure

Equivalence: Three views are equivalent. Given one, derive the others.

Why PGMs are Central to Generative Modeling :

Native probabilistic language:

- Built from probability theory first principles
- Every component has direct probabilistic semantics
- No translation from functions to distributions needed

Explicit structural representation:

- Conditional dependencies: which variables influence others
- Causal relationships, hierarchical organization
- Temporal dynamics, relational structure
- Graph *is* the structure

Operations preserve probability axioms:

- Products maintain proper normalization
- Marginalization and conditioning well-defined
- Conditional independence enables tractable probability

Explicit generative story:

- Sample from roots (no parents)
- Sample children conditioned on parents
- Follow edges = follow generative process

Principled use of all operations:

- Products: factorization reduces complexity
- Mixtures: latent variables create multimodality
- Composition: hierarchical structure
- All maintain proper probability measures

Latent Variable Models: An Important Class :

Definition: PGMs with observed variables X and unobserved (latent) variables Z.

Generative process:

$$Z_i \sim P(Z;\theta) \tag{77}$$
$$X_i \sim P(X|Z_i;\phi) \tag{78}$$

Joint and marginal:

$$P(X,Z) = \prod_i P(Z_i)P(X_i|Z_i) \tag{79}$$
$$P(X) = \int P(X,Z)\,dZ = \int \prod_i P(Z_i)P(X_i|Z_i)\,dZ \tag{80}$$

Why LVMs matter:

- Model hidden structure (clusters, topics, representations)
- Compression: low-dimensional encodings
- Interpretability: semantic factors
- Structure discovery: unsupervised learning

Example: Gaussian Mixture Model

- Latent z: categorical cluster assignment
- $P(z = k) = \alpha_k$
- $P(x|z = k) = \mathcal{N}(x|\mu_k, \sigma_k^2)$
- Marginal: $P(x) = \sum_k \alpha_k \mathcal{N}(x|\mu_k, \sigma_k^2)$

The Computational Challenge:
This is where PGMs reveal their weakness despite conceptual elegance.
Intractable marginalization:

$$P(X) = \int P(X, Z)\, dZ \tag{81}$$

- High-dimensional latent spaces
- Complex dependencies
- Exponential configurations

Intractable inference:

$$P(Z|X) = \frac{P(X, Z)}{P(X)} = \frac{P(X, Z)}{\int P(X, Z)\, dZ} \tag{82}$$

- Posterior computation requires marginalization
- Needed for learning (EM's E-step)

Classical solution: EM Algorithm
Alternate between:

- **E-step**: Compute $p(z|x; \theta^{(t)})$ or $\mathbb{E}_{p(z|x)}[\log P(x, z)]$
- **M-step**: $\theta^{(t+1)} = \arg\max_\theta \mathbb{E}_{p(z|x;\theta^{(t)})}[\log P(x, z; \theta)]$

Properties:

- Monotonic improvement
- Converges to local maximum
- Elegant but doesn't scale
- Still requires tractable E-step

Limitations:

- Slow convergence
- Local optima
- Requires tractable posteriors (often not available)
- Doesn't scale to complex, high-dimensional models

This computational bottleneck motivates the synergy with neural networks.

5.3.2 Neural Networks: Powerful Function Approximators

What neural networks are:

- Function approximators: composition of parameterized transformations
- Not natively probabilistic
- Universal approximation: can represent arbitrarily complex functions
- Gradient-based learning via backpropagation

What they bring:

- **Expressiveness**: Approximate complex conditionals $p(x|z)$ without hand-specified forms
- **Learning from data**: Discover patterns, hierarchical features
- **Flexibility**: Adapt architecture to data modality (CNNs, Transformers, GNNs, U-Nets)
- **Scalability**: Efficient optimization, automatic differentiation

Probabilistic adaptation challenge:

- **Normalization**: Must ensure valid probability (softmax, explicit normalization, partition functions)
- **No automatic guarantees**: Can produce invalid probabilities if unconstrained
- **Implicit structure**: Dependencies learned in weights, not explicit
- **Less interpretable**: Black box nature

5.3.3 The Synergy: Navigating the Structure-Capacity Spectrum

The complementarity:

PGMs provide	NNs provide
Probabilistic scaffolding	Expressive parameterization
Explicit structure	Learning capacity
Principled operations	Flexible approximation
Interpretable dependencies	Universal approximation
Structure (factorization)	Capacity (representation)

Modern generative models combine both: PGM structure + NN components

The Essential Choice: Structure vs Capacity Every modeling decision involves navigating this trade-off:

- **More structure** (PGM-heavy): Factorize, impose independence, explicit relationships
- **More capacity** (NN-heavy): Let networks learn everything, minimal assumptions

Example 1: Encoder - Graph to Latent
Task: Given graph X, compute encoding $p(Z|X)$.
Extreme 1 - Pure Structure (PGM):

- Factorize: $p(Z|X) = \prod_i p(z_i|\text{parents}_{\text{PGM}}(z_i), X)$
- Each conditional simple (Gaussian, categorical)

- Heavy assumptions about independence
- Few parameters, interpretable
- Needs less data

Extreme 2 - Pure Capacity (NN):

- Single GNN: Graph $X \xrightarrow{\text{GNN}} Z$
- One monolithic function
- No explicit independence assumptions
- Many parameters, black box
- Needs more data, more flexible

Spectrum in between:

- Some PGM factorization where each factor is NN-parameterized
- $p(Z|X) = p(z_1|X)p(z_2|X, z_1)$ where each conditional uses GNN
- Balance structure and capacity

Example 2: Decoder - Latent to Graph Task: Given latent Z, generate graph $p(X|Z)$.

Extreme 1 - Pure Structure (Autoregressive):

- Heavy factorization: $p(X|Z) = p(x_1|Z)p(x_2|x_1, Z) \cdots p(x_n|x_{1:n-1}, Z)$
- Sequential generation, explicit dependencies
- GraphRNN approach
- Each step can use NN, but structure is explicit

Extreme 2 - Pure Capacity (One-shot):

- Single NN: $Z \xrightarrow{\text{NN}} X$
- Generate entire graph at once
- No explicit sequential structure
- Maximum flexibility, minimal assumptions

Spectrum in between:

- Partial factorization with NN components
- E.g., generate nodes then edges, each step NN-based

Architecture Choice Emerges from Data Structure :

Key insight: When data has structure, choose NN architecture accordingly:

- **Images (2D grid)**: CNNs exploit translation equivariance, local structure
- **Sequences (1D)**: RNNs/Transformers handle temporal dependencies
- **Graphs (irregular)**: GNNs via message passing over arbitrary topology
- **Point clouds**: PointNet, permutation-invariant architectures
- **3D volumes**: 3D CNNs, U-Nets with skip connections

Examples of synergy:

- **VAEs**: PGM structure $p(x, z) = p(x|z)p(z)$ + NN encoder $q_\phi(z|x)$, NN decoder $p_\theta(x|z)$
- **Normalizing Flows**: PGM change-of-variables + NN transformations
- **Diffusion**: PGM Markov chain + NN denoising at each step
- **Autoregressive**: PGM factorization $\prod_i p(x_i|x_{<i})$ + NN conditionals
- **GraphVAE**: PGM latent structure + GNN encoder/decoder for graph data

Navigating the Trade-off :

Choose based on:

- **Knowledge**: How much structure do you know/trust?
- **Data**: How much data available? (More data → can afford more capacity)
- **Interpretability**: Need explainable structure? (More structure → more interpretable)
- **Flexibility**: Unknown patterns to learn? (More capacity → more flexible)
- **Compute**: Computational constraints? (Structure can be more efficient)

5.3.4 A Note on Probabilistic Circuits

The operations we've introduced—mixtures (sums) and products—form the algebraic basis of **probabilistic circuits** Darwiche [2009], Choi et al. [2020], a formal framework that deserves more attention than it currently receives in modern AI research.

What are Probabilistic Circuits? Directed acyclic graphs where:

- **Sum nodes**: Represent mixtures (weighted combinations)
- **Product nodes**: Represent factorization (independence assumptions)
- **Leaf nodes**: Input distributions or deterministic transformations
- **Circuit structure**: Defines composition of operations

Key Properties: Under certain structural constraints (smoothness, decomposability), probabilistic circuits guarantee:

- Tractable exact inference
- Tractable marginal computation
- Polynomial-time queries

Why It Matters—And Why It's Underappreciated: Probabilistic circuits provide a **principled algebraic framework** for combining the operations we've discussed, with formal guarantees about tractability. Yet modern deep generative models (VAEs, GANs, Flows, Diffusion) largely developed independently, without drawing on this formalism.

This represents a gap between:

- **Classical probabilistic modeling**: Emphasizes tractability, formal guarantees
- **Modern deep learning**: Emphasizes scalability, empirical performance

There may be valuable insights in bridging these traditions—using circuit-theoretic principles to design tractable deep architectures, or scaling circuit methods with neural parameterization.

Forward: We focus on PGMs and neural networks as the dominant frameworks in modern generative AI, but probabilistic circuits offer an alternative perspective worth exploring for readers interested in tractable probabilistic reasoning.

5.4 Summary and Forward Look

Key insights:

- **Operations**: Transformations (reshape), mixtures (increase variance), products (decrease variance), composition
- **Two frameworks**:
  - PGMs: Native probabilistic language, explicit structure, built for modeling structure
  - NNs: Universal function approximators, implicit learning, built for capacity
- **The essential trade-off**: Structure vs Capacity
  - Manage complexity through factorization (structure) OR approximation (capacity)
  - Not binary choice - continuous spectrum
  - Modern models combine both
- **Synergy addresses computational challenges**:
  - LVMs conceptually elegant but computationally intractable
  - NNs solve PGM computational bottleneck
  - PGM scaffolding + NN components = modern deep generative models
- **Data structure guides architecture**: CNNs for grids, GNNs for graphs, Transformers for sequences

Forward look:

Different generative model categories represent different points on the structure-capacity spectrum:

- **Autoregressive** (AR): Heavy structure via factorization, NN conditionals
- **GANs**: Minimal structure, maximum capacity
- **VAEs**: Balanced - latent structure + neural components
- **Flows**: Constrained capacity (invertibility), mathematical structure

- **Diffusion**: Temporal structure (Markov chain) + neural denoising

These models don't just differ in implementation details - they represent fundamentally different philosophies about managing complexity through structure versus capacity.

We explore these model families in detail in subsequent sections, understanding each through this unified lens.

Part II

Categories of Generative Models

6 Early Deep Generative Models

6.1 Pioneering Architectures (1990-2009)

Several foundational architectures explored deep probabilistic modeling before the modern deep learning era:

- **Sigmoid Belief Network** Neal [1990]: Early architecture using sigmoid units with stochastic binary activations (1990)
- **Helmholtz Machine** Dayan et al. [1995]: Introduced recognition model concept for approximate inference (1995)
- **Non-linear Gaussian Belief Networks** Frey et al. [1999]: Extended linear Gaussian models with non-linear transformations (1999)
- **Restricted Boltzmann Machine + Contrastive Divergence** Hinton [2002]: Introduced the contrastive divergence (CD) algorithm for efficiently training Restricted Boltzmann Machines and products of experts models (2002)
- **Deep Belief Network** Hinton et al. [2006]: Stacked restricted Boltzmann machines enabling layer-wise pretraining (2006)
- **Deep Boltzmann Machine** Salakhutdinov and Hinton [2009]: Fully undirected model with multiple hidden layers

6.2 Bottlenecks from the "Control Variance" Ideology

The pioneering architectures (1990-2009) shared a common assumption: **variance must be exactly computed and contained within closed-form expressions.**

6.2.1 For Undirected Models (RBM, DBM)

- Required: $p(\mathbf{x};\theta) = \frac{1}{Z(\theta)} \exp(-E(\mathbf{x};\theta))$
- Computing $Z(\theta) = \sum_{\mathbf{x}} \exp(-E(\mathbf{x};\theta))$ intractable
- This made **both learning and inference intractable**
- Learning needs $\frac{\partial \log Z}{\partial \theta}$ (model expectations)
- Inference needs marginalization over exponentially many states

6.2.2 For Directed Models (Helmholtz Machine, DBN)

- No partition function, but $p(\mathbf{h}|\mathbf{x})$ still intractable
- Explaining away creates complex posterior dependencies
- Required approximate inference (Wake-Sleep, layer-wise pretraining)

6.2.3 Brittleness of Analytical Methods

- Each new model required **pages of mathematical derivations**
- Custom inference algorithms for each architectural variant
- No general-purpose learning framework
- Limited flexibility and extensibility

6.2.4 Inadequacy of Approximations

- Mean field approximations often **too restrictive**
- Assumed independence where strong dependencies exist
- Resulted in poor posterior approximations
- Quality-scalability tradeoff: better approximations even more expensive

6.2.5 The Real Problem

These weren't just computational bottlenecks. The classical framework tried to **capture all variance in a single closed-form expression** – treating variance as a quantity to measure exactly, bound tightly, and contain within explicit probability distributions.

For deep variance spaces (like vision: one concept $\rightarrow$ infinite valid instantiations), this is **fundamentally the wrong approach**.

7 Autoregressive Models

7.1 The Task: Sequential Generation

- Some data has natural ordering: text, code, time series, speech
- Each element depends on what came before
- "The cat sat on the ___" → next word depends on context
- Temporal data: tomorrow's weather depends on today's

7.2 The Natural Approach

- Generate left-to-right (or past-to-future)
- Use context to predict next element
- Fundamental question: $p(\text{next} \mid \text{previous})$
- Chain rule does the rest: $p(\mathbf{x}) = \prod p(x_i \mid \mathbf{x}_{<i})$

7.3 What Makes This Natural

- Sequential structure is intrinsic to the data
- Not imposed (unlike raster-scanning images)
- Order matters: "dog bites man" $\neq$ "man bites dog"
- Prediction task is well-defined

7.4 Core Principle

- Follow chain rule of probability: $p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i \mid \mathbf{x}_{<i})$
- Make no independence assumptions beyond ordering
- Universal: can represent any distribution over ordered variables

7.5 Nature of Data: Sequences

- Autoregressive models are designed for **sequential data**
- Natural ordering exists or can be imposed:
  - Natural: time series, natural language, code, audio
  - Imposed: images (raster scan order)
- The sequential structure motivates:
  - The autoregressive factorization (respects temporal/spatial order)
  - Neural architectures designed for sequences (RNNs, Transformers)
- Inductive bias: architecture matches data structure

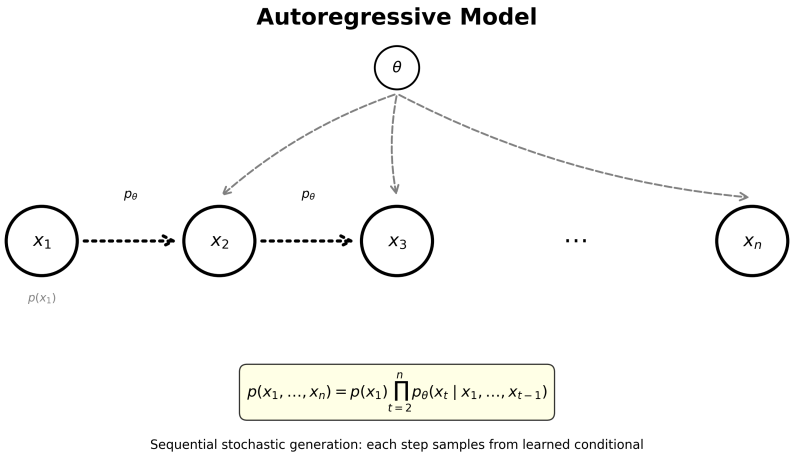

Figure 2: Autoregressive Model structure. The joint distribution factorizes as $p(x_1, \ldots, x_n) = p(x_1) \prod_{t=2}^{n} p_\theta(x_t \mid x_1, \ldots, x_{t-1})$. Each variable x_t is generated by sampling from the conditional distribution given all previous variables (dotted arrows indicate stochastic sampling). Parameters θ govern the learned conditional distributions. This sequential generation is the foundation for models like PixelCNN, WaveNet, and GPT. The autoregressive factorization ensures tractable likelihood computation and exact sampling, at the cost of sequential (non-parallel) generation.

7.6 Classical Autoregressive Models (Pre-Neural)

- N-gram language models: $p(w_i \mid w_{i-1}, \ldots, w_{i-n+1})$ as frequency tables
- Markov chains: $p(x_t \mid x_{t-1}, \ldots, x_{t-k})$ with fixed-order dependencies
- AR(p) time series models in statistics [**Yule (1927), Box and Jenkins (1970)**]
- Conditional probability tables for discrete variables
- Key limitation: exponential parameters OR restrictive fixed-context assumptions

7.7 Modeling Considerations

- Conditional distributions per variable type:
  - Binary: Bernoulli($\rho_\theta(\cdot)$)
  - Categorical: Cat(softmax$_\theta(\cdot)$)
  - Real-valued: $\mathcal{N}(\mu_\theta(\cdot), \sigma_\theta(\cdot))$

7.8 What Neural Networks Add

- Capacity: model long-range dependencies without exponential parameters
- Flexibility: learn which past context matters (vs fixed windows)
- Function approximation: $\rho_\theta(\cdot) = \sigma(\text{MLP}_\theta(\mathbf{x}_{<i}))$ instead of lookup tables
- Optimization: gradient-based learning via backpropagation
- But: same fundamental principle (chain rule factorization)

7.9 Neural Autoregressive Models: Foundational References

- **FVSBN (Fully Visible Sigmoid Belief Network)** - Frey [1998]
  - **Breakthrough:** First neural AR with *linear number of parameters* (single layer)
  - AR factorization + weight sharing = O(d) parameters instead of exponential
  - Eliminated intractable inference of general Sigmoid Belief Networks
- **LSTM** - Hochreiter & Schmidhuber [1997] Hochreiter and Schmidhuber [1997]
  - **Breakthrough:** Solved vanishing gradient problem via *state/memory and gating mechanisms*
  - Enabled learning long-range dependencies in sequences
- **NADE** - Larochelle & Murray [2011] Larochelle and Murray [2011]
  - **Breakthrough:** Extended FVSBN's *linear parameterization to deep architectures* (with hidden layers)
  - First tractable neural autoregressive density estimator with depth
  - Showed neural networks + AR = explicit likelihood without intractable partition function
- **PixelRNN/PixelCNN** - van den Oord et al. [2016] van den Oord et al. [2016b]
  - **Breakthrough:** Showed AR works for high-dimensional images via *masked convolutions*
  - Proved sequential generation viable for 2D spatial data (respecting causal ordering)
- **WaveNet** - van den Oord et al. [2016] van den Oord et al. [2016a]
  - **Breakthrough:** Raw audio waveform generation with *dilated causal convolutions*
  - Exponentially growing receptive fields without exponential parameters
- **Attention is All You Need** - Vaswani et al. [2017] Vaswani et al. [2017]
  - **Breakthrough:** *Self-attention mechanism* replaces recurrence entirely
  - Parallel processing + global context without sequential bottleneck
- **GPT** - Radford et al. [2018] Radford et al. [2018]
  - **Breakthrough:** *Large-scale pre-training + task fine-tuning* paradigm using Transformers
  - Demonstrated generative AR pre-training enables strong performance across diverse downstream tasks

7.10 Task Performance Analysis

MODELING ✓✓

- Makes no assumptions beyond ordering
- Universal: can represent any distribution
- Tradeoff: requires choosing variable order

LEARNING ✓✓

- Exact maximum likelihood (no approximation)
- Simple gradient-based optimization
- Efficient: parallelizable across data and sequence positions

SCORING ✓✓

- Exact $p(\mathbf{x})$ via chain rule: $\log p(\mathbf{x}) = \sum_i \log p(x_i \mid \mathbf{x}_{<i})$
- Efficient: parallelizable across sequence
- No intractable partition function

SAMPLING ×

- Sequential: must generate one step at a time
- Slow: $O(T)$ for sequence length T
- Cannot parallelize generation

INFERENCE ×

- No latent variables to infer
- Cannot learn unsupervised representations
- No explicit "understanding" separate from generation

7.11 The Fundamental Tradeoff

- **Wins**: Perfect scoring, exact learning, no approximations
- **Sacrifices**: Slow sequential generation, no latent structure
- **When to choose**: When scoring matters more than generation speed, or when latent structure not needed

8 Variational Autoencoders (VAE)

8.1 Beyond Sequential Structure

- AR works beautifully for sequences (text, time series)
- But what about data without natural ordering?
- Images: no inherent left-to-right, top-to-bottom order
- Want to model images, but sequential generation feels forced

8.2 The Need for Latent Representations

- Natural question: What if data is generated from hidden factors?
- Image of face: generated from underlying attributes (age, expression, lighting)
- These factors = latent variables
- Data = observable, latent = unobservable causes

8.3 Two Complementary Goals

1. **Generation**: Sample from $p(\mathbf{x})$ to create new data
2. **Inference**: Given $\mathbf{x}$, infer latent $\mathbf{z}$ (representation learning)

- Learn meaningful representations, not just generate
- Enable: interpolation, manipulation, understanding

8.4 Why This Matters

- Latent space can be semantically meaningful
- "Face space": smooth changes in $\mathbf{z} \rightarrow$ smooth changes in attributes
- Useful beyond generation: clustering, visualization, transfer learning
- AR gives no latent structure – just sequential dependencies

8.5 The Challenge

- Need to model $p(\mathbf{x})$ and $p(\mathbf{z}|\mathbf{x})$ jointly
- Inference $p(\mathbf{z}|\mathbf{x})$ typically intractable
- Solution: variational inference with neural networks

8.6 Core Principle

- Latent variable model: $p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$
- Inference problem: posterior $p(\mathbf{z}|\mathbf{x})$ intractable
- Solution: approximate posterior with variational distribution $q_\phi(\mathbf{z}|\mathbf{x})$
- Optimize Evidence Lower Bound (ELBO) jointly for generation and inference

8.7 Nature of Data: General, Architecture Adapts

- VAEs work for any data type with suitable likelihood $p(\mathbf{x}|\mathbf{z})$
- Architecture choice depends on data structure:
  - Images: Convolutional encoder/decoder
  - Text: Recurrent or Transformer encoder/decoder
  - Molecules: Graph neural networks
- Latent space typically continuous (unlike discrete for AR models)
- Inductive bias: encoder/decoder architectures match data structure

8.8 Classical Latent Variable Models (Pre-Neural)

- Factor Analysis: linear Gaussian model
  - $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  - $\mathbf{x}|\mathbf{z} \sim \mathcal{N}(\mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \boldsymbol{\Sigma})$
  - EM algorithm for learning
- Gaussian Mixture Models: discrete latent variable
  - $z \sim \text{Cat}(\boldsymbol{\pi})$
  - $\mathbf{x}|z \sim \mathcal{N}(\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z)$
- Classical variational inference: mean-field approximation
  - $q(\mathbf{z}) = \prod_i q_i(z_i)$ with simple factors
  - Coordinate ascent optimization
- Key limitations: linear mappings, per-datapoint optimization expensive

8.9 What Neural Networks Add

- Nonlinear encoder/decoder: $\boldsymbol{\mu}_\theta(\mathbf{z})$ and $\boldsymbol{\Sigma}_\theta(\mathbf{z})$ as neural networks
- Amortized inference: single encoder $q_\phi(\mathbf{z}|\mathbf{x})$ shared across all data
  - Classical: optimize q separately for each $\mathbf{x}$ (expensive)
  - VAE: learn one encoder network, apply to any $\mathbf{x}$ (fast)
- Reparameterization trick: $\mathbf{z} = \boldsymbol{\mu}_\phi(\mathbf{x}) + \boldsymbol{\sigma}_\phi(\mathbf{x}) \odot \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  - Enables low-variance gradient estimates
  - Makes stochastic variational inference practical
- Universal approximation of complex posteriors and likelihoods
- But: same fundamental principle (variational inference for latent variable model)

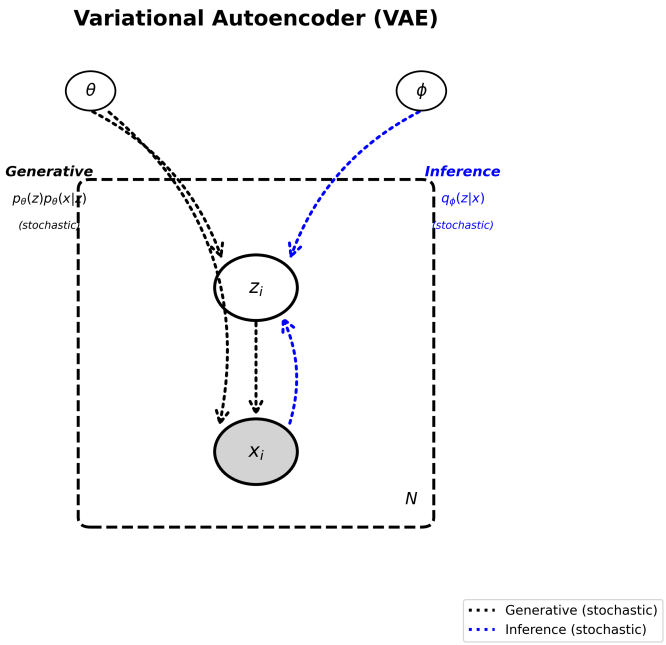

Figure 3: Plate diagram for Variational Autoencoder (VAE). The generative model (dotted black arrows) consists of the stochastic prior $p_\theta(z)$ and likelihood $p_\theta(x|z)$ parameterized by decoder θ. The inference model (dotted blue arrows) consists of the stochastic approximate posterior $q_\phi(z|x)$ parameterized by encoder ϕ. Both processes are stochastic (shown by dotted lines), unlike the deterministic transformations in normalizing flows. The plate represents N independent samples. Shaded node x_i is observed data, unshaded node z_i is the latent variable.

8.10 The VAE Framework

- Generative model:
  - Prior: $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$
  - Likelihood: $p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\boldsymbol{\mu}_\theta(\mathbf{z}), \boldsymbol{\Sigma}_\theta(\mathbf{z}))$
  - Marginal: $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$ (intractable)
- Variational distribution:
  - $q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}_\phi(\mathbf{x}), \text{diag}(\boldsymbol{\sigma}_\phi(\mathbf{x})))$
  - Encoder/Recognition network/Inference network

8.11 Evidence Lower Bound (ELBO)

- $\log p_\theta(\mathbf{x}) \geq \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \text{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))$
- First term: reconstruction likelihood
- Second term: KL regularization (keeps q close to prior)
- Tight when $q_\phi(\mathbf{z}|\mathbf{x}) = p_\theta(\mathbf{z}|\mathbf{x})$

8.12 VAE Learning Algorithm

- Sample minibatch $\{\mathbf{x}_i\}$ from data

- Encode: compute $\boldsymbol{\mu}_\phi(\mathbf{x}_i)$, $\boldsymbol{\sigma}_\phi(\mathbf{x}_i)$
- Sample: $\mathbf{z}_i = \boldsymbol{\mu}_\phi(\mathbf{x}_i) + \boldsymbol{\sigma}_\phi(\mathbf{x}_i) \odot \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- Decode: compute $p_\theta(\mathbf{x}_i|\mathbf{z}_i)$
- Compute ELBO and gradients wrt θ, ϕ
- Update parameters with gradient ascent
- Iterate

8.13 Stochastic Variational Inference

- Expectation $\mathbb{E}_{\mathbf{z}\sim q}[\cdot]$ approximated by Monte Carlo:
  - $\mathbb{E}[f(\mathbf{z})] \approx \frac{1}{L}\sum_{l=1}^{L} f(\mathbf{z}_l)$ where $\mathbf{z}_l \sim q$
- Often $L = 1$ (single sample) sufficient due to reparameterization
- Minibatch gradient estimates for scalability

8.14 Reparameterization Trick

Problem: Cannot backprop through sampling $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$
The issue is computing:

$$\nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[f(\mathbf{z})]$$

The distribution itself depends on ϕ (appears in the subscript):

- Cannot simply swap ∇_ϕ and $\mathbb{E}_{q_\phi}$ because q_ϕ depends on ϕ
- Monte Carlo approximation: $\mathbb{E}_{q_\phi}[f(\mathbf{z})] \approx \frac{1}{L}\sum_{l=1}^{L} f(\mathbf{z}^{(l)})$ where $\mathbf{z}^{(l)} \sim q_\phi$
- Taking gradient: $\nabla_\phi \left[\frac{1}{L}\sum_{l=1}^{L} f(\mathbf{z}^{(l)})\right]$
- **The problem**: Each $\mathbf{z}^{(l)}$ was drawn from q_ϕ but appears as a constant in the sum
- Changing ϕ affects *which samples* we get, but this dependency is hidden in the sampling operation
- The gradient doesn't "see" how ϕ affects the sampling distribution—only how it affects f at fixed samples
- **High variance and biased gradients** result from naive Monte Carlo differentiation

Solution: rewrite as deterministic function of auxiliary noise

- $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- $\mathbf{z} = g_\phi(\boldsymbol{\epsilon}, \mathbf{x}) = \boldsymbol{\mu}_\phi(\mathbf{x}) + \boldsymbol{\sigma}_\phi(\mathbf{x}) \odot \boldsymbol{\epsilon}$
- Take expectation over $\boldsymbol{\epsilon}$ instead of $\mathbf{z}$
- Enables low-variance gradient estimates
- Gradients flow through encoder

8.15 Connection to Classical Autoencoders

- Classical autoencoder: deterministic encoder/decoder
  - $\mathbf{z} = f_\phi(\mathbf{x})$ (deterministic encoding)
  - $\hat{\mathbf{x}} = g_\theta(\mathbf{z})$ (deterministic decoding)
  - Minimize reconstruction error $\|\mathbf{x} - \hat{\mathbf{x}}\|^2$
- Cannot generate: no distribution over $\mathbf{z}$, random $\mathbf{z}$ doesn't decode meaningfully
- VAE adds:
  - Probabilistic encoder: distribution $q_\phi(\mathbf{z}|\mathbf{x})$
  - Prior $p(\mathbf{z})$: enables sampling $\mathbf{z} \sim p(\mathbf{z})$ for generation
  - Regularization: KL term prevents "holes" in latent space

8.16 Variational Autoencoders: Foundational References

- **Deep Latent Gaussian Models**
  - Rezende, Mohamed & Wierstra [2014] - "Stochastic Backpropagation and Approximate Inference" Rezende et al. [2014]
  - **Breakthrough:** Hierarchical deep latent Gaussian models with multiple stochastic layers
- **Neural Networks for Variational Approximation**
  - Kingma & Welling [2014] - "Auto-Encoding Variational Bayes" Kingma and Welling [2014]
  - **Breakthrough:** Parameterizing variational posterior $q(\mathbf{z}|\mathbf{x})$ with neural networks (recognition networks)
- **Amortized Variational Inference**
  - Kingma & Welling [2014] - "Auto-Encoding Variational Bayes" Kingma and Welling [2014]
  - **Breakthrough:** Single inference network for all datapoints, not per-datapoint optimization
  - Scalable inference through parameter sharing
- **Reparameterization Trick**
  - Kingma & Welling [2014] AND Rezende et al. [2014] Kingma and Welling [2014], Rezende et al. [2014]
  - **Breakthrough:** $\mathbf{z} = \mu + \sigma \odot \epsilon$ enables backpropagation through stochastic sampling
  - Made gradient-based learning of deep generative models practical
- **VQ-VAE (Vector Quantized Variational Autoencoder)**
  - van den Oord et al. [2017] van den Oord et al. [2017]
  - **Breakthrough:** *Discrete latent codes via vector quantization* instead of continuous Gaussian latents
  - Fundamental shift from continuous to discrete latent representations
  - Enabled bridging VAE and autoregressive models

8.17 Task Performance Analysis

MODELING ✓✓

- Flexible: can model complex distributions via latent variables
- Structured: explicit latent space for representations
- Continuous latent space enables interpolation

LEARNING ✓

- Stable gradient-based optimization
- Amortized inference: encoder shared across data
- Tradeoff: approximation gap (ELBO $\leq \log p(\mathbf{x})$)

SCORING ∼

- Lower bound: can compute ELBO (lower bound on $\log p(\mathbf{x})$)
- Cannot compute exact $p(\mathbf{x})$ (intractable marginal)
- Approximate: importance sampling or annealed importance sampling

SAMPLING ✓✓

- Fast: sample $\mathbf{z} \sim p(\mathbf{z})$, decode $\mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z})$
- Parallel: can generate many samples simultaneously
- Quality often not great (until recent improvements)

INFERENCE ✓✓

- Explicit encoder $q_\phi(\mathbf{z}|\mathbf{x})$
- Fast: single forward pass through encoder
- Learns meaningful latent representations
- Enables: interpolation, arithmetic in latent space, conditional generation

8.18 The Fundamental Tradeoff

- **Wins**: Learns latent representations, stable training, fast inference and sampling, principled probabilistic framework
- **Sacrifices**: Approximate likelihood (ELBO gap), samples sometimes blurry
- **When to choose**: When learning representations matters, or when stable training and principled inference are priorities

9 Generative Adversarial Networks

9.1 The Question After VAE

- VAE works: learns latent representations, stable training
- But: samples often blurry, ELBO approximation gap
- Quality not quite there for high-fidelity images
- Natural question: Do we really need explicit $p(\mathbf{x})$?

9.2 An Alternative Philosophy

- VAE: maximize likelihood, learn $p(\mathbf{x})$ explicitly
- GAN: abandon likelihood entirely
- Goal: just generate samples that look real
- Learn via comparison: can discriminator tell real from fake?

9.3 The Adversarial Idea

- Don't model $p(\mathbf{x})$ directly
- Instead: two networks compete
- Generator: fool the discriminator
- Discriminator: detect fakes
- Competition drives improvement

9.4 Why This Was Exciting (2014-2020)

- Generated sharp, high-quality images
- StyleGAN: photorealistic faces
- State-of-art for images for several years
- Showed likelihood-free learning possible

9.5 The Tradeoff

- Wins: impressive sample quality
- Sacrifices: can't compute $p(\mathbf{x})$, unstable training, mode collapse
- No inference (no encoder by default)

9.6 An Evolutionary Branch

- Important historical approach
- Coexisted with explicit likelihood models
- Proved adversarial learning viable
- But: the explicit likelihood lineage (VAE $\rightarrow$ Flows $\rightarrow$ Diffusion) proved more fruitful
- Diffusion now dominates where GAN once ruled

9.7 Why Cover This

- Historical significance
- Different philosophy worth understanding
- Shows alternative paths in design space
- Illustrates tradeoffs: likelihood vs sample quality, stability vs speed

9.8 Core Principle

- Learn distributions by distinguishing real from generated samples
- Likelihood-free: no explicit density model $p(\mathbf{x})$
- Adversarial game: generator vs discriminator
- Based on two-sample test: can we tell if two sample sets come from same distribution?

9.9 Nature of Data: Originally Images, Now General

- GANs pioneered for image generation (where they excelled until ∼2021)
- Architecture adapts to data type:
  - Images: Convolutional networks (DCGAN, StyleGAN)
  - Text: Transformers with discrete outputs (challenging)
  - Audio, video: temporal convolutions
- Inductive bias: discriminator architecture matches data structure

9.10 Classical Two-Sample Testing (Pre-Neural)

- Statistical hypothesis testing: given $S_1 \sim P$ and $S_2 \sim Q$, is $P = Q$?
- Classical tests: Kolmogorov-Smirnov, Maximum Mean Discrepancy
- Any classifier can serve as discriminator (decision trees, SVMs, logistic regression)
- Key limitation: limited capacity to distinguish complex distributions

9.11 What Neural Networks Add

- Capacity: both generator and discriminator can model complex distributions
- Differentiability: enables minimax optimization via gradients
- Expressivity: learns subtle differences between real and generated data
- But: same fundamental principle (adversarial two-sample comparison)

9.12 The GAN Framework

- Data samples: $S_1 = \{\mathbf{x} \sim p^*\}$ from true distribution
- Model samples: $S_2 = \{\mathbf{x} \sim q_\theta\}$ from generator
- Discriminator $D_\phi(\mathbf{x})$: learns to distinguish real from fake
- Generator $G_\theta(\mathbf{z})$: learns to fool discriminator
- Minimax game: $\min_\theta \max_\phi V(q_\theta, p^*)$

9.13 Discriminator Objective

- $V(q_\theta, p^*) = \mathbb{E}_{\mathbf{x}\sim p^*}[\log D_\phi(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D_\phi(G_\theta(\mathbf{z})))]$
- Maximize: make discriminator better at distinguishing
- Optimal discriminator: $D^*(\mathbf{x}) = \frac{p^*(\mathbf{x})}{p^*(\mathbf{x})+q_\theta(\mathbf{x})}$

9.14 Generator Objective

- $\min_\theta \max_\phi V(q_\theta, p^*)$
- Minimize: make generator better at fooling discriminator
- At equilibrium: $q_\theta = p^*$ (generator matches data distribution)

9.15 GAN Learning Algorithm

- Sample m training points from data $\mathcal{D}$
- Sample m noise vectors, generate synthetic points
- Update discriminator ϕ using gradient ascent
- Update generator θ using gradient descent
- Repeat until convergence (or divergence...)

9.16 Generative Adversarial Networks: Foundational References

- **GAN (Generative Adversarial Networks)**
  - Goodfellow et al. [2014] Goodfellow et al. [2014]
  - **Breakthrough:** *Adversarial training framework* – generator vs discriminator game
  - Likelihood-free learning through implicit density modeling
  - Demonstrated viability of learning generative models without explicit density computation
- **DCGAN (Deep Convolutional GAN)**
  - Radford, Metz & Chintala [2015] Radford et al. [2016]
  - **Breakthrough:** *Convolutional architectures* made GANs work for images at scale
  - Established architectural best practices (batch normalization, transposed convolutions)
  - Made training stable enough for practical use
- **Wasserstein GAN (WGAN)**
  - Arjovsky, Chintala & Bottou [2017] Arjovsky et al. [2017]
  - **Breakthrough:** *Wasserstein distance* provides meaningful loss and training stability
  - Theoretical foundation for understanding GAN training dynamics
  - Addressed mode collapse and vanishing gradients
- **BiGAN (Bidirectional GAN)**
  - Donahue, Krähenbühl & Darrell [2016] Donahue et al. [2017]

- **Breakthrough:** Showed *representation learning possible with GANs* despite lacking explicit encoder
- Bidirectional mapping: both generation ($\mathbf{z} \to \mathbf{x}$) and inference ($\mathbf{x} \to \mathbf{z}$)
- Extended GAN framework to include learned representations

9.17 Task Performance Analysis

MODELING ✓✓

- Flexible: makes no distributional assumptions
- Implicit density: no explicit $p(\mathbf{x})$ needed
- Can model complex high-dimensional distributions

LEARNING ×

- Unstable optimization (minimax game)
- Mode collapse: generator ignores parts of data distribution
- Requires careful hyperparameter tuning
- No convergence guarantees

SCORING ××

- Cannot compute $p(\mathbf{x})$ - no explicit density
- Cannot evaluate likelihood
- Hard to compare models quantitatively

SAMPLING ✓✓

- Fast: single forward pass $G_\theta(\mathbf{z})$
- Parallel: can generate many samples simultaneously
- High-quality samples (when training succeeds)

INFERENCE ×

- No encoder by default
- Cannot infer latent code for given $\mathbf{x}$
- BiGAN/ALI add inference but complicate training

9.18 The Fundamental Tradeoff

- **Wins**: Fast parallel sampling, high-quality images (when stable), no likelihood needed
- **Sacrifices**: Cannot score, unstable training, mode collapse, no inference
- **When to choose**: When sample quality matters more than density estimation, and you can tolerate training instability

9.19 Looking Back: Why GANs Ultimately Weren't Enough

- Generated impressive samples (2014-2020)
- But persistent challenges: mode collapse, training instability
- Something fundamentally mismatched for images
- Led to key question: What makes images fundamentally different?

9.19.1 The Diagnosis (In Retrospect)

Two characteristics GANs didn't fully address:

Variance Depth

- Vision = richer variance wellspring than language
- One concept → infinite visual instantiations
- Must resolve enormous detail
- Single-step generation (even adversarial) insufficient capacity

Spatial Structure

- Images are 2D spatial, not sequential
- GANs do respect this (parallel generation)
- But: single-step still limited

9.19.2 The Path Forward

- Need: compositional capacity for deep variance
- Need: parallel processing for 2D structure
- This diagnosis led to: Flows, then Diffusion
- The explicit likelihood lineage proved more fruitful

10 Normalizing Flows

10.1 Addressing the Image Challenge

- After GAN: diagnosed what images fundamentally need
- **Variance depth** → compositional capacity
- **Spatial structure** → parallel processing
- Question: Can we address BOTH?

10.2 The Flows Approach: Deterministic Composition

- Many invertible transformations composed
- Parallel: whole-image operations (respects 2D structure)
- Compositional: builds capacity gradually (handles variance)
- Exact likelihood: no GAN instability

10.3 The Natural Question from VAE

- VAE uses reparameterization: $\mathbf{z} = \boldsymbol{\mu}_\phi(\mathbf{x}) + \boldsymbol{\sigma}_\phi(\mathbf{x}) \odot \boldsymbol{\epsilon}$ (one transformation)
- Chain rule and backprop already handle composition
- Question: Why stop at one? Stack K transformations!
- $\mathbf{z}_0 \sim p(\mathbf{z}_0) \to \mathbf{z}_1 = f_1(\mathbf{z}_0) \to \mathbf{z}_2 = f_2(\mathbf{z}_1) \to \ldots \to \mathbf{x} = f_K(\ldots f_1(\mathbf{z}_0))$
- Change of variables: $\log p(\mathbf{x}) = \log p_0(\mathbf{z}_0) - \sum_i \log |\det(\partial f_i / \partial \mathbf{z}_{i-1})|$

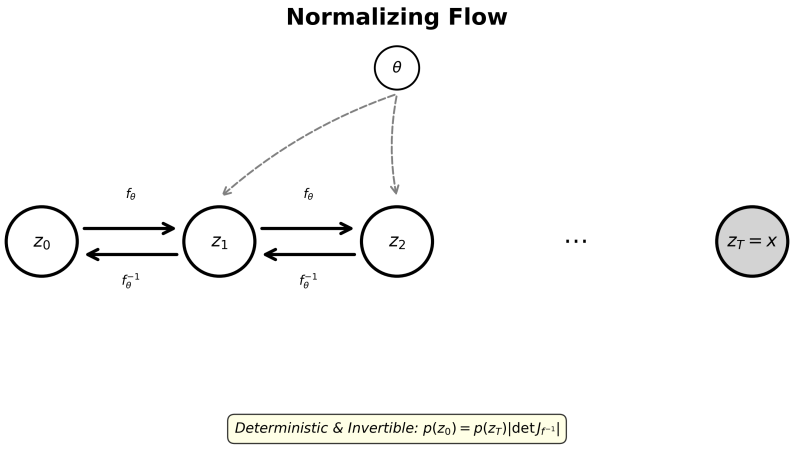

Figure 4: Normalizing Flow structure. A sequence of deterministic, invertible transformations f_θ maps latent variable $z_0 \sim p(z_0)$ to data $x = z_T$. Each transformation is bijective, allowing exact likelihood computation via change of variables: $p(x) = p(z_0)|\det J_{f^{-1}}|$. Bidirectional arrows show invertibility—both forward generation ($z_0 \to x$) and exact inference ($x \to z_0$) are tractable. The final state $z_T = x$ is observed data (shaded).

10.4 But: The Devil in the Details

- Need Jacobian determinant each step
- O(d^3) for arbitrary transformations
- 800K dimensions → completely intractable
- Entire field exists to solve this via clever architectures

10.4.1 Devil #1: The Jacobian Determinant

- Need $\det(\partial f/\partial \mathbf{z})$ for each transformation
- For d-dimensional data: naive computation is $\mathbf{O}(d^3)$
- Example: 256×256 RGB image → $d \approx 200{,}000$
  - Single Jacobian: $(200{,}000)^3 \approx 8 \times 10^{15}$ operations
  - **Completely intractable!**
- This is why VAE stops at one transformation: can avoid Jacobian via ELBO
- **This single constraint drives ALL normalizing flow architecture design**

Why Jacobian is Expensive:

- Matrix determinant: O(d^3) via LU decomposition
- Need this for EVERY transformation in the sequence
- For K layers: $K\times$ O(d^3)
- No way around it for arbitrary neural networks

10.4.2 Understanding the Jacobian

The Jacobian isn't just a computational obstacle—it reveals the fundamental modeling choice.

What Jacobian Elements Mean: Each element $J[i,j] = \partial x_i/\partial z_j$ measures: **How much does changing input dimension j affect output dimension i?**

This captures the **interaction/dependence** between dimensions:

- $J[i,j] \neq 0$: Dimensions i and j interact
- $J[i,j] = 0$: Dimensions i and j are independent
- Full Jacobian (all elements nonzero): Models all cross-dimensional interactions

The Structure-Capacity Trade-off: Reducing Jacobian complexity = Making assumptions about data structure

- **Full Jacobian**: Maximum expressiveness, models arbitrary interactions, O(d^3) - intractable
- **Structured Jacobian**: Assumes limited interactions, sacrifices expressiveness, O(d) - tractable

This is the structure-capacity trade-off in action:

- **Capacity**: Ability to model complex cross-dimensional dependencies
- **Structure**: Assumptions about which dimensions interact
- **The exchange**: We gain tractability by constraining which interactions the model can learn

Constructive Design Principle: This understanding enables principled architecture design:

- **Question**: What assumptions about dimensional interactions are reasonable for my data?
- **Autoregressive**: Assumes causal ordering—dimension i independent of dimensions $j > i$
- **Coupling**: Assumes dimensions can be split into independent groups
- **Sparsity**: Assumes most dimension pairs don't interact

The choice encodes inductive bias about data structure.

10.4.3 Devil #2: Ensuring Invertibility

- f must be invertible to compute $\mathbf{z}_0 = f^{-1}(\mathbf{x})$
- Most neural networks are NOT invertible
- Even if theoretically invertible, numerical instability
- Can't just stack arbitrary layers and hope

10.4.4 Devil #3: Expressivity vs Tractability

The fundamental tension:

- **Expressivity** means modeling many cross-dimensional interactions
- More interactions → fuller Jacobian → higher computational complexity
- Maximum expressivity (all dimensions interact) → full Jacobian → $O(d^3)$
- Tractability requires sparse/structured Jacobian → restricted interactions

This is the structure-capacity trade-off:

- Capacity: Model complex dependencies between all dimensions
- Structure: Assume limited interaction patterns (ordering, groupings, sparsity)
- Cannot maximize both: must choose based on data and task

Normalizing flow design is fundamentally about:

- What interaction structure to impose (architectural constraint)
- How many layers to compensate (composition for capacity)
- Which inductive biases match the data domain

10.5 How Normalizing Flows Solve These Devils

The entire field exists to make the Jacobian tractable through clever architecture:

10.5.1 Solution 1: Coupling Layers (RealNVP, Glow)

- **Idea**: Structure function so Jacobian is triangular
- Split dimensions: $\mathbf{z} = [\mathbf{z}_A, \mathbf{z}_B]$
- Transform only half:
  - $\mathbf{z}'_A = \mathbf{z}_A$ (unchanged)
  - $\mathbf{z}'_B = \mathbf{z}_B \odot \exp(s(\mathbf{z}_A)) + t(\mathbf{z}_A)$
- Jacobian is block triangular:

$$\frac{\partial f}{\partial \mathbf{z}} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ * & \text{diag}(\exp(s)) \end{bmatrix}$$

- Determinant: $\det = \exp(\sum s(\mathbf{z}_A)) \rightarrow \mathbf{O(}d\mathbf{)}$ not $O(d^3)$!
- Invertible: $\mathbf{z}_B = (\mathbf{z}'_B - t(\mathbf{z}_A)) \odot \exp(-s(\mathbf{z}_A))$
- **Price**: Only transforms half the dimensions per layer

10.5.2 Solution 2: Autoregressive Flows (MAF, IAF)

- **Idea**: Transform each dimension conditioned on previous
- Jacobian is triangular by construction
- Diagonal elements: $\partial z'_i / \partial z_i$ easy to compute
- Determinant: product of diagonal $\rightarrow \mathbf{O(}d\mathbf{)}$
- **Price**: Sequential computation
  - MAF: parallel density, sequential sampling
  - IAF: sequential density, parallel sampling

10.5.3 Solution 3: Continuous Normalizing Flows (Neural ODEs)

- **Idea**: Replace discrete layers with continuous dynamics
- $d\mathbf{z}/dt = f_\theta(\mathbf{z}, t)$
- Instantaneous change of variables:
  - $d(\log p)/dt = -\text{trace}(\partial f_\theta / \partial \mathbf{z})$
- Trace not determinant! $\rightarrow \mathbf{O(}d^2\mathbf{)}$ with Hutchinson estimator $\rightarrow \mathbf{O(}d\mathbf{)}$
- **Price**: ODE integration (more compute), harder to invert

10.6 The Architectural Constraints

All flows must satisfy:

1. **Tractable Jacobian**: $O(d)$ or $O(d^2)$, never $O(d^3)$
2. **Invertibility**: Explicit inverse or guaranteed numerically stable
3. **Expressivity**: Need many layers to compensate for individual simplicity

10.7 Nature of Data: Architecture Adapts

- Images: coupling layers with convolutional structure
- Sequences: autoregressive within flow layers
- General: choice depends on density vs sampling priority
- Inductive bias: coupling structure determines interaction patterns

10.8 Classical Transformations (Pre-Neural)

- Box-Muller: uniform $\to$ Gaussian (explicit, invertible)
- Linear: $\mathbf{z} \to \mathbf{A}\mathbf{z} + \mathbf{b}$ ($\det = |\det(\mathbf{A})|$)
- Elementwise monotonic: exp, log, sigmoid
- Change of variables formula known for centuries
- Key limitation: simple functions insufficient for complex data

10.9 What Neural Networks Add

- Parameterize complex invertible functions
- Learn transformations from data
- Universal approximation through composition
- But: must respect computational constraints
- **Not arbitrary NNs** – carefully constrained architectures

10.10 Flow Architectures in Detail

RealNVP (Real-valued Non-Volume Preserving)

- Coupling layers with affine transformations
- Multi-scale: progressively reduce dimensions
- Checkerboard/channel-wise masking patterns
- Each layer: $O(d)$ Jacobian, explicit inverse

Glow

- Adds invertible 1×1 convolutions
- Actnorm: activation normalization

- Multi-scale with split/squeeze operations
- State-of-art for image generation (pre-diffusion)

MAF (Masked Autoregressive Flow)

- Fast density estimation: $p(\mathbf{x})$ computable in parallel
- Slow sampling: must generate sequentially
- Use case: density estimation, anomaly detection

IAF (Inverse Autoregressive Flow)

- Fast sampling: can generate in parallel
- Slow density: $p(\mathbf{x})$ computed sequentially
- Use case: generative modeling with VAE-style training

Neural Spline Flows

- Monotonic splines for transformations
- More flexible than affine
- Still $\mathrm{O}(d)$ Jacobian via coupling

Continuous Normalizing Flows (FFJORD)

- ODE integration for transformation
- Trace estimation via Hutchinson
- Unlimited "depth" (continuous time)

10.11 Flow Learning Algorithm

- Sample $\mathbf{x}$ from data
- **Inverse pass**: compute $\mathbf{z}_0 = f_K^{-1} \circ \ldots \circ f_1^{-1}(\mathbf{x})$
  - Each step: compute $\log|\det(\partial f_i/\partial \mathbf{z}_i)|$
  - Accumulate: $\log p(\mathbf{x}) = \log p_0(\mathbf{z}_0) - \sum_i \log|\det|$
- Maximize $\log p(\mathbf{x})$ via gradient ascent
- **Forward pass** (generation): $\mathbf{z}_0 \sim p_0 \to \mathbf{x} = f_K \circ \ldots \circ f_1(\mathbf{z}_0)$

10.12 Why This Works Despite Devils

- Clever architectures: $\mathrm{O}(d)$ Jacobians instead of $\mathrm{O}(d^3)$
- Many simple layers: compensate for individual restrictions
- Architectural innovation: entire subfield devoted to new flow designs
- Still expensive: typically slower than VAEs, but exact likelihood worth it

10.13 Connection to VAE

- VAE: one transformation, approximate posterior (avoid Jacobian)
- Flow: K transformations, exact density (pay Jacobian cost cleverly)
- View VAE as 1-layer flow with approximate inference
- Flow removes approximation at computational cost

10.14 Normalizing Flows: Foundational References

- **NICE (Non-linear Independent Components Estimation)**
  - Dinh, Krueger & Bengio [2014] Dinh et al. [2015]
  - **Breakthrough:** *Coupling layers* with tractable Jacobian determinant
  - Solved O(d^3) determinant computation problem through architectural design
  - Enabled exact likelihood training for deep generative models
- **RealNVP (Real-valued Non-Volume Preserving)**
  - Dinh, Sohl-Dickstein & Bengio [2016] Dinh et al. [2017a]
  - **Breakthrough:** *Affine coupling layers* – more expressive than NICE's additive couplings
  - Demonstrated flows work for high-dimensional image generation
  - Established flows as competitive generative models
- **Glow**
  - Kingma & Dhariwal [2018] Kingma and Dhariwal [2018]
  - **Breakthrough:** *Invertible 1×1 convolutions* as learnable permutations
  - Demonstrated flows competitive with GANs for high-quality image synthesis
  - Combined actnorm, invertible convolutions, and affine coupling
- **MAF (Masked Autoregressive Flow)**
  - Papamakarios, Pavlakou & Murray [2017] Papamakarios et al. [2017]
  - **Breakthrough:** *Autoregressive structure enables triangular Jacobians*
  - Each dimension depends on all previous dimensions (highly expressive)
  - O(d) determinant via diagonal product
  - Trade-off: fast density evaluation, slow sampling (sequential generation)
- **IAF (Inverse Autoregressive Flow)**
  - Kingma et al. [2016] Kingma et al. [2016]
  - **Breakthrough:** Inverse of MAF – same autoregressive structure, reversed computation
  - O(d) determinant computation
  - Trade-off: slow density evaluation, fast sampling (parallel generation)
  - Particularly useful when fast sampling is priority
- **Neural ODEs / Continuous Normalizing Flows**

- Chen et al. [2018] Chen et al. [2018]
- **Breakthrough:** *Continuous-time flows using ordinary differential equations*
- Trace estimator instead of determinant: O(d) computation
- From discrete transformations to continuous dynamics
- Enabled unrestricted architectures (no invertibility constraints)

10.15 Task Performance Analysis

MODELING ✓✓

- Exact: no approximation gap
- Flexible: universal approximation with enough layers
- Constrained: architecture must satisfy invertibility + tractable Jacobian

LEARNING ✓

- Exact maximum likelihood
- Stable gradients
- Tradeoff: architectural constraints, more layers needed

SCORING ✓✓

- Exact $p(\mathbf{x})$ via change of variables
- O(d) per layer if architecture chosen correctly
- No intractability (unlike VAE marginal)

SAMPLING ✓

- Forward transformation from $\mathbf{z}_0$
- Fast for IAF-style, slow for MAF-style
- Quality: high-fidelity

INFERENCE ✓✓

- Exact inverse: $\mathbf{z}_0 = f^{-1}(\mathbf{x})$
- Deterministic latent code
- Meaningful latent space

10.16 The Fundamental Tradeoffs

- **Wins**: Exact likelihood, exact inference, no approximation
- **Sacrifices**:
  - Architectural constraints (can't use arbitrary NNs)
  - More layers needed (individual layers restricted)
  - Computational cost (though O(d) not O(d^3))
- **When to choose**: Exact density critical (anomaly detection, compression, density estimation), can afford architectural constraints

10.17 Why This Matters

- Not just "VAE with more layers"
- Fundamental computational challenge solved through architecture
- Entire research area: designing flows with tractable Jacobians
- Understanding tradeoffs essential for choosing/designing flows

10.18 What Flows Achieve

- Compositional capacity ✓
- Parallel processing ✓
- Exact likelihood ✓
- Stable training ✓

10.19 But: Is This Enough Variance Capacity?

- Vision's variance extraordinarily deep
- Deterministic composition helps
- Question ahead: could we capture variance even better with stochasticity?

11 Diffusion Models

11.1 After Flows: Need Even More Variance Capacity

11.1.1 Flows Succeeded

- Addressed both characteristics: composition + parallel
- Exact likelihood, stable training
- Architectural innovations solved Jacobian problem

11.1.2 But: Vision's Variance Is Extraordinarily Deep

- Flows help via many transformations
- Still deterministic at each step
- Natural question: can we model variance even better?

11.1.3 The Insight: Embrace Stochasticity

- Flows: many deterministic transformations
- What if: many STOCHASTIC transformations?
- Add randomness at every step
- Compose randomness itself
- Connects to core principle: "variance is wellspring of creation"

11.1.4 From Deterministic to Stochastic Composition

- Flows: $\mathbf{z}_0 \to f_1 \to \mathbf{z}_1 \to f_2 \to \ldots$ (deterministic)
- Diffusion: $\mathbf{z}_0 \to (f_1 + \text{noise}) \to \mathbf{z}_1 \to (f_2 + \text{noise}) \to \ldots$ (stochastic)
- Not just more steps, but fundamentally different at each step

11.2 Two Paths to Diffusion Models

Path 1: Via Normalizing Flows (already traversed)

- VAE: one transformation (stochastic reparameterization)
- NF: many transformations (deterministic, invertible)
- DM: many transformations (stochastic instead of deterministic)
- Insight: Embrace variance in composition, avoid Jacobian constraints

Path 2: Via Denoising Autoencoders (discovered for free!)

- VAE: learn encoder + decoder, stochastic latent $\mathbf{z}$
- DAE: fix corruption (add noise), learn only reversal
  - $\tilde{\mathbf{x}} = \mathbf{x}+$ noise $\to$ reconstruct $\mathbf{x}$
  - One corruption step, one denoising

- DM: compose multiple corruption + denoising steps
  - $\mathbf{x}_0 \to \mathbf{x}_1 \to \ldots \to \mathbf{x}_T$ (forward: fixed noise schedule)
  - $\mathbf{x}_T \to \mathbf{x}_{T-1} \to \ldots \to \mathbf{x}_0$ (reverse: learned denoising)
- Insight: Fix forward process, learn reverse only

Two paths, same destination

- Path 1: Compose first (NF), then add stochasticity
- Path 2: Fix corruption first (DAE), then compose
- DM sits at intersection: multiple stochastic transformations with fixed forward

11.3 Core Principle

- Forward process (diffusion): gradually corrupt data with Gaussian noise (fixed)
- Reverse process (denoising): learn to reverse corruption step by step
- Train: predict noise at each timestep
- Generate: start from pure noise, iteratively denoise

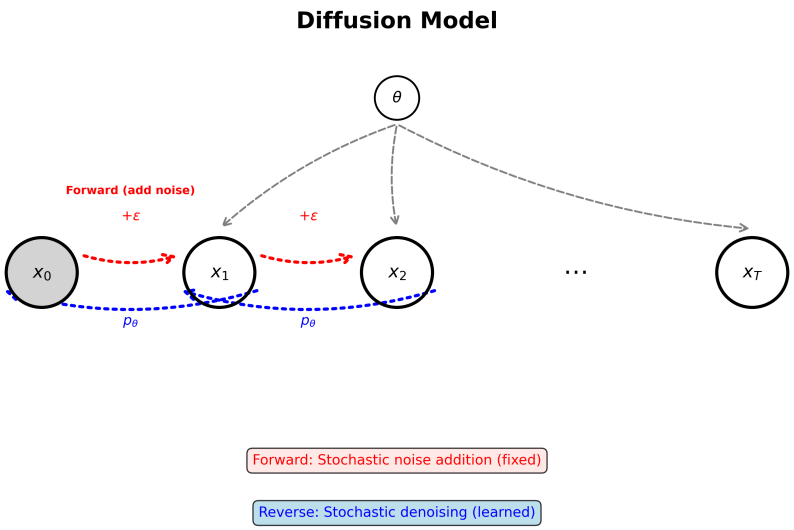

Figure 5: Diffusion Model structure. Forward process (red, dotted arrows): Fixed stochastic noise addition gradually corrupts data x_0 into pure noise $x_T \sim \mathcal{N}(0, I)$. Reverse process (blue, dotted arrows): Learned stochastic denoising $p_\theta(x_{t-1}|x_t)$ gradually removes noise to generate data. Unlike flows, both directions are stochastic (shown by dotted lines and noise symbols $+\epsilon$), and the reverse process must be learned. The initial state x_0 is observed data (shaded).

11.4 Nature of Data: General, Architecture Adapts

- Works for images, audio, video, molecules
- Architecture for denoising network matches data:
  - Images: U-Net (convolutional, skip connections)
  - Audio: WaveNet-style
  - General: must handle noisy input at all noise levels
- Inductive bias: denoising architecture matches data structure

11.5 Classical Foundations (Pre-Neural)

- Non-equilibrium thermodynamics: Jarzynski equality, forward-reverse processes
- Langevin dynamics: sampling via gradient + noise
  - $d\mathbf{z} = \nabla \log p(\mathbf{z})dt + \sqrt{2}dW$
  - Brownian motion with drift toward high probability
- Score matching (Hyvärinen 2005): estimate $\nabla \log p(\mathbf{x})$ without partition function
- Annealed Langevin: add noise schedule for better mixing
- Key insight: can sample by following score (gradient of log density)

11.6 What Neural Networks Add

- Learn denoising function at ALL noise levels
  - Classical: hand-crafted or simple denoising
  - DM: $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ learns noise prediction conditioned on timestep
- Universal approximation: handle complex data distributions
- U-Net architecture: preserve spatial information via skip connections
- Conditioning: can condition on labels, text, etc.
- But: same principle (gradual corruption + reversal)

11.7 The Forward Process (Fixed)

- $q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$
- Noise schedule $\{\beta_t\}$: typically linear or cosine
- Markov chain: $\mathbf{x}_0 \to \mathbf{x}_1 \to \ldots \to \mathbf{x}_T$
- Closed form: $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$
  - $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$
- At $t = T$: $\mathbf{x}_T \approx \mathcal{N}(\mathbf{0}, \mathbf{I})$ (pure noise)

11.8 The Reverse Process (Learned)

- $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$
- Learn to denoise at each step
- Parameterization: predict noise $\boldsymbol{\epsilon}$
  - $\boldsymbol{\mu}_\theta = \frac{1}{\sqrt{1-\beta_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right)$
- Start from noise, iteratively denoise to data

11.9 Training Objective

- Maximize ELBO (like VAE)
- Simplifies to: $\mathbb{E}[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\|^2]$
- At each step t: predict the noise that was added
- Simple: just train denoising network on noisy images

11.10 Training Algorithm

- Sample $\mathbf{x}_0$ from data
- Sample t uniformly from $\{1, \ldots, T\}$
- Sample noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- Create noisy: $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$
- Predict: $\hat{\boldsymbol{\epsilon}} = \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$
- Loss: $\|\boldsymbol{\epsilon} - \hat{\boldsymbol{\epsilon}}\|^2$
- Update θ

11.11 Sampling Algorithm

- Sample $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- For $t = T$ to 1:
  - Predict noise: $\hat{\boldsymbol{\epsilon}} = \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$
  - Compute mean: $\boldsymbol{\mu}_\theta$
  - Sample: $\mathbf{x}_{t-1} = \boldsymbol{\mu}_\theta + \sigma_t \mathbf{z}$ where $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- Return $\mathbf{x}_0$

11.12 THE DEVILS: Why This Costs

11.12.1 Devil #1: Slow Sampling

- Need T steps (typically $T = 1000$)
- Each step: full network forward pass
- vs GAN/VAE/Flow: 1 forward pass
- 1000× slower generation!
- Trade quality for speed (fast samplers: DDIM, 50 steps)

11.12.2 Devil #2: Learning Denoising at All Noise Levels

- Network must handle $\mathbf{x}_t$ for all $t \in \{1, \ldots, T\}$
- Different noise levels = different tasks
- Conditioning on t essential
- Must learn T different denoisings

11.12.3 Devil #3: Training Stability vs Sample Quality

- More steps (larger T) $\rightarrow$ better samples but slower
- Noise schedule design matters (linear vs cosine)
- Balancing act: training ease vs generation quality

11.13 Solutions to Devils

- DDIM: deterministic sampling, fewer steps
- Fast samplers: distillation, consistency models
- Score-based continuous: replace discrete steps with SDEs
- Latent diffusion: diffuse in latent space (smaller dimension)

11.14 Connection to Score Matching

- Denoising $\approx$ score estimation
- $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \approx -\sqrt{1-\bar{\alpha}_t}\nabla \log p(\mathbf{x}_t)$
- Diffusion = learned annealed Langevin dynamics
- Multiple noise levels = annealing schedule

11.15 Diffusion Models: Foundational References

- **Deep Unsupervised Learning using Nonequilibrium Thermodynamics**
  - Sohl-Dickstein et al. [2015]
  - **Breakthrough:** *Diffusion probabilistic models* – gradual noising with learned reverse process
  - Connected thermodynamics, Langevin dynamics, and generative modeling
  - Established fixed forward process + learned reverse framework
- **DDPM (Denoising Diffusion Probabilistic Models)**
  - Ho et al. [2020]
  - **Breakthrough:** *Simplified denoising objective* made diffusion practical
  - Reweighted variational bound for stable training
  - Demonstrated high-quality image synthesis competitive with GANs
- **Score-Based Generative Modeling**

  - Song and Ermon [2019], Song et al. [2020]
  - **Breakthrough:** *Score matching + Langevin dynamics* perspective on diffusion
  - Showed learning $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ equivalent to denoising
  - Unified score-based models and diffusion models mathematically

- **DDIM (Denoising Diffusion Implicit Models)**
  - Song et al. [2021a]
  - **Breakthrough:** *Non-Markovian sampling process* enables deterministic generation
  - Fast sampling (10-50 steps instead of 1000) without retraining
  - Interpolation and manipulation in latent space

- **Classifier Guidance**
  - Dhariwal and Nichol [2021] "Diffusion Models Beat GANs on Image Synthesis"
  - **Breakthrough:** *Conditional generation via classifier gradients*
  - Guides diffusion process using $\nabla_{\mathbf{x}} \log p(y|\mathbf{x})$
  - Trade-off: sample quality vs diversity via guidance scale

- **Classifier-Free Guidance**
  - Ho and Salimans [2022]
  - **Breakthrough:** *Conditioning without separate classifier*
  - Joint training of conditional and unconditional models
  - Simplified training, better quality than classifier guidance

11.16 Task Performance Analysis

MODELING ✓✓

- Flexible: can model complex distributions
- Fixed forward, learn reverse
- No architectural constraints like flows

LEARNING ✓✓

- Stable training (simple MSE loss)
- No adversarial dynamics (unlike GAN)
- Gradient flow well-behaved

SCORING ✓

- Can compute (approximate) likelihood via ELBO
- Score function $\nabla \log p(\mathbf{x})$ available via network
- Not as direct as AR or flows

SAMPLING ×

- Slow: T forward passes (typically 1000)

- Fast samplers help but still $\gg$ 1 pass
- Quality excellent when patient

INFERENCE $\sim$

- No explicit encoder like VAE
- Can invert via DDIM (deterministic path)
- Score function gives gradient information

11.17 Fundamental Tradeoff

- **Wins**: State-of-art sample quality (2021-now), stable training, no architectural constraints
- **Sacrifices**: Slow generation (1000 steps), no fast latent inference
- **When**: Sample quality paramount, can afford generation time, or use fast samplers

11.18 Why Two Paths Matter

- Path 1 (via NF): emphasizes composition + stochasticity
- Path 2 (via DAE): emphasizes fixed corruption + composition
- Both valid, both insightful
- Shows DM as natural convergence point in design space
- Not arbitrary – conceptually motivated from multiple directions

11.19 Latent Diffusion Models: Solving the Speed Problem

11.19.1 The Problem: Diffusion Models are Slow

- DMs achieve state-of-art quality
- But: generation takes seconds to minutes (vs milliseconds for GAN/VAE)
- 1000 denoising steps in high-dimensional space
- **Why so slow? Two distinct reasons:**

Reason 1: Each Step is Expensive

- Operating in pixel space: $512 \times 512 \times 3 \approx 800$K dimensions
- Each denoising step: full network forward pass in high-d space
- Computational cost scales with dimensionality
- 1000 steps $\times$ 800K dimensions = expensive

Reason 2: Need Many Steps

- Pixel space: must maintain local coherence
- Each step makes small adjustment
- Operating at low level of abstraction
- Need $\sim$1000 small steps to transform noise $\rightarrow$ coherent image

11.19.2 The Idea: Transform to Lower-Dimensional Latent Space

- We already know how to compress: VAE!
- Encoder: $\mathbf{x} \to \mathbf{z}$ ($512 \times 512 \times 3 \to 64 \times 64 \times 4$)
- Diffusion: operate in $\mathbf{z}$-space instead of $\mathbf{x}$-space
- Decoder: $\mathbf{z} \to \mathbf{x}$

11.19.3 Result: One Solution Addresses BOTH Problems!

Benefit 1: Each Step Cheaper (Addresses Reason 1)

- Dimensional reduction: 800K $\to$ 16K dimensions ($\sim$48$\times$ reduction)
- Each denoising network pass: $\sim$48$\times$ cheaper
- Guaranteed computational speedup

Benefit 2: Fewer Steps Needed (Addresses Reason 2)

- Latent space dimensions semantically meaningful
- One step in $\mathbf{z}$-space = larger semantic change
- Operating at higher level of abstraction
- Each step accomplishes more
- Typically need only $\sim$50 steps instead of $\sim$1000 ($\sim$20$\times$ reduction)

Why Semantic Efficiency Happens

- VAE learned perceptually meaningful compression
- Latent dimensions capture high-level features
- Moving in feature space vs pixel space
- Higher abstraction $\to$ problem becomes easier
- Not just smaller, but structurally better space to work in

Combined Impact

- $\sim$48$\times$ cheaper per step (dimensional)
- $\sim$20$\times$ fewer steps (semantic)
- Total: $\sim$1000$\times$ speedup
- Real-time generation becomes possible

11.19.4 The Composition: VAE + Diffusion

- VAE encoder E: $\mathbf{x} \to \mathbf{z}$ (compress)
- Diffusion in $\mathbf{z}$-space:
  - Forward: $\mathbf{z}_0 \to \mathbf{z}_1 \to \ldots \to \mathbf{z}_T$
  - Reverse: $\mathbf{z}_T \to \ldots \to \mathbf{z}_0$ (learned)
- VAE decoder D: $\mathbf{z} \to \mathbf{x}$ (decompress)

11.19.5 Training

- Pre-train VAE separately (or use existing)
- Freeze VAE, train diffusion in latent space
- Loss: standard denoising objective on $\mathbf{z}$
- Optional: joint fine-tuning

11.19.6 Generation

- Sample $\mathbf{z}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ in latent space
- Denoise: $\mathbf{z}_T \to \mathbf{z}_0$ ($\sim$50 steps)
- Decode: $\mathbf{x} = D(\mathbf{z}_0)$
- Can condition on text (CLIP, T5 embeddings)

11.19.7 Task Performance

- MODELING ✓✓: flexible + efficient
- LEARNING ✓✓: stable, faster
- SCORING $\sim$: approximate (VAE gap)
- SAMPLING ✓✓: $\sim$1000$\times$ faster
- INFERENCE ✓: VAE encoder

11.19.8 Tradeoff

- Gain: Speed (both sources), manageable compute
- Cost: VAE approximation, decoder artifacts
- When: Speed critical, quality/approximation acceptable

11.19.9 Design Principle: Right Level of Abstraction

- Analyze problem: why is it hard?
- Multiple bottlenecks $\to$ need solution addressing all
- Transform problem space itself
- Latent space: not just smaller, but better structured
- Dimensional reduction + semantic efficiency compound

11.19.10 Why This Matters: Problem-Driven Composition

- Started with problem (speed)
- Analyzed structure (two reasons)
- Found elegant solution (addresses both)
- Used existing tool (VAE) in new way
- **Models compose to solve problems neither solves alone**
- VAE: meaningful compression
- Diffusion: high-quality generation
- Together: fast high-quality generation

11.20 Latent Diffusion Models: Foundational References

Conceptual Framework: Latent diffusion operates in a latent space $\mathcal{Z}$ rather than data space $\mathcal{X}$, with generation $z \to x$ via $p(x|z;\theta)$. The parameters θ may be learned (via reconstruction objectives) or unlearned/frozen (leveraging pre-trained representations).

- **High-Resolution Image Synthesis with Latent Diffusion Models**
  - Rombach et al. [2022] Rombach et al. [2022]
  - **Breakthrough:** *Diffusion in learned compressed latent space with trained decoder*
  - Perceptual compression via VAE: encoder $x \to z$, decoder $z \to x$
  - Diffusion operates in compressed $\mathcal{Z}$, not pixel space $\mathcal{X}$
  - Decoder $p(x|z;\theta)$ learned via reconstruction loss
  - Computational efficiency: reduced dimensionality, maintains perceptual quality
  - Enabled high-resolution generation at practical computational cost
- **DALL-E 2 (Hierarchical Text-Conditional Image Generation)**
  - Ramesh et al. [2022] Ramesh et al. [2022]
  - **Breakthrough:** *Diffusion in frozen pre-trained semantic space*
  - Prior network: text $\to$ CLIP image embedding (latent space)
  - Decoder: CLIP embedding $\to$ pixels via diffusion
  - $p(x|z;\theta)$ uses frozen CLIP space (no decoder training)
  - Leverages pre-trained semantic structure without additional decoder optimization
  - Two-stage: prior learns text $\to$ image embedding, decoder unprojects to pixels

11.20.1 Closure of Model Categories

- Started: AR (no latent), GAN (implicit), VAE (one stochastic layer)
- Built: Flows (many deterministic), Diffusion (many stochastic)
- Composed: Latent Diffusion (VAE + Diffusion solving complementary problems)
- **Not isolated models but connected schema**
- Understanding structure enables intelligent composition
- Design space has logic: analyze problems, combine solutions

11.21 Score-Based Models: The Continuous Perspective

11.21.1 Connection to Diffusion

- Already covered: Langevin dynamics, score matching in classical foundations
- Natural question: Is there a deeper connection?
- Answer: Denoising diffusion and score matching are the same thing

11.21.2 The Mathematical Connection

Score function

- Definition: $\nabla \log p(\mathbf{x})$ (gradient of log density)
- Points toward higher probability regions
- Classic use: Langevin sampling

Denoising = Score Estimation

- Diffusion model learns: $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$
- Score function: $\nabla \log p(\mathbf{x}_t)$
- Mathematical equivalence: $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \approx -\sqrt{1-\bar{\alpha}_t}\nabla \log p(\mathbf{x}_t)$
- Learning to predict noise = learning score function

Why This Matters

- Diffusion: "gradually denoise corrupted data"
- Score-based: "follow gradient of log density"
- Same process, different language
- Understanding score connects to broader statistical literature

11.21.3 Two Framings of Same Idea

Discrete Time (DDPM - what we covered)

- Timesteps: $t \in \{1, 2, \ldots, T\}$
- Forward: discrete noise schedule
- Reverse: learned denoising steps
- T typically 1000

Continuous Time (Score-Based SDE)

- Time: $t \in [0, T]$ continuous
- Forward: Stochastic Differential Equation (SDE)
  - $d\mathbf{x} = f(\mathbf{x}, t)dt + g(t)dw$
  - f: drift, g: diffusion coefficient, dw: Brownian motion
- Reverse: reverse-time SDE using score

  - $d\mathbf{x} = [f(\mathbf{x}, t) - g(t)^2 \nabla \log p(\mathbf{x}, t)]dt + g(t)d\bar{w}$
- Infinite timesteps (continuous)

Unification (Song & Ermon, Yang Song et al.)

- Framework: covers both discrete and continuous
- DDPM = discretization of continuous SDE
- Score matching = continuous perspective on denoising
- Flexible noise schedules (variance preserving, variance exploding)
- Mathematical elegance

11.21.4 Practical Implications

Same model, different view:

- Train denoising network (DDPM approach)
- Interpret as score estimator (score-based view)
- Use either discrete or continuous sampling

Advantages of continuous view:

- Flexible ODE/SDE samplers
- Controllable noise schedules
- Theoretical analysis cleaner
- Connection to broader mathematical frameworks

11.21.5 Not a Different Model

- Score-based models $\neq$ new model category
- Different mathematical framing of diffusion
- Vocabulary: when you hear "score-based generative models" $\rightarrow$ continuous-time diffusion
- When you hear "score matching" $\rightarrow$ training objective equivalent to denoising

11.21.6 Key Takeaway

- Diffusion models and score-based models: same underlying idea
- Discrete vs continuous framing
- Denoising = score estimation (mathematically equivalent)
- One framework, multiple perspectives

12 Multimodal Models: Generation Across Modalities

12.1 The Natural Question

- Built models for single modalities:
  - Text: AR (GPT)
  - Images: Diffusion, GANs, VAEs
- Natural next step: **Generate one modality conditioned on another**
- Most prominent: Text $\rightarrow$ Image

12.2 Core Challenge: The Semantic Gap

- Different modalities = different representations
  - Text: discrete tokens
  - Images: continuous pixels
- Need **shared semantic space**
- How to align "a photo of a cat" with actual cat images?

12.3 Nature of Data: Paired Modalities

- Requires paired examples: (image, caption)
- Web-scale datasets available
- Weak supervision: alt-text, captions

12.4 Classical Foundations (Pre-Neural)

- Canonical Correlation Analysis: find correlated projections
- Multi-view learning: different "views" of same content
- Key insight: align via shared statistical structure
- Limitation: linear, hand-crafted features

12.5 What NNs Add

- Learned shared embedding spaces
- Contrastive learning: align via similarity
- Cross-attention: query one modality with another
- Scale to billions of pairs
- Same principle: alignment via statistical structure

12.6 THE DEVILS

12.6.1 Devil #1: Creating Shared Semantic Space

- Fundamentally different representations
- How to define "semantic similarity" across modalities?
- Need embedding where "cat" text near cat images
- Solution: Contrastive learning

12.6.2 Devil #2: Conditioning Architecture

- Have generative model (e.g., Diffusion)
- Have conditioning signal (text embedding)
- Where/how to combine?
- Cross-attention is key mechanism

12.6.3 Devil #3: Controlling Generation Strength

- How strongly should text guide generation?
- Too weak: ignores prompt
- Too strong: unrealistic artifacts
- Solution: Guidance mechanisms

12.7 Solution 1: Contrastive Alignment (CLIP)

12.7.1 The Idea

- Learn joint embedding space
- Image encoder: $I \to \mathbf{z}_I$
- Text encoder: $T \to \mathbf{z}_T$
- Maximize similarity for matching pairs
- Minimize for non-matching pairs

12.7.2 Why This Works

- No pixel reconstruction needed
- Learn via comparison
- Scales to billions of web pairs
- Emergent semantic understanding

12.7.3 Result

- Text and images embedded in same space
- Semantically similar = close in embedding
- Enables cross-modal conditioning

12.8 Solution 2: Cross-Attention Conditioning

12.8.1 The Mechanism

- Text tokens as keys/values
- Image features as queries
- Attention: which text tokens relevant for each image region?
- Multiple layers throughout network
- Learned interaction patterns

12.8.2 Why Cross-Attention

- Flexible: image regions attend to relevant text
- Not just early concatenation
- Bidirectional information flow
- Standard in modern text-to-image

12.9 Putting Together: Text-to-Image Generation

12.9.1 Architecture (Stable Diffusion style)

- Text encoder: CLIP (text $\rightarrow$ embeddings)
- Diffusion model with cross-attention
  - Image path: noisy latent features
  - Text path: conditioning embeddings
  - Cross-attention: fuse at multiple layers
- VAE decoder: latent $\rightarrow$ pixels

12.9.2 Training

- Dataset: (image, caption) pairs
- Encode text: $\mathbf{z}_{\text{text}}$
- Encode image: $\mathbf{z}_{\text{image}}$ (VAE)
- Add noise schedule: $\mathbf{z}_t$
- Learn denoising: $\boldsymbol{\epsilon}_\theta(\mathbf{z}_t, t, \mathbf{z}_{\text{text}})$
- Condition on text throughout

12.9.3 Generation

- Input: text prompt
- Encode: $\mathbf{z}_{\text{text}}$
- Sample noise: $\mathbf{z}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- Denoise conditioned on text
- Decode: $\mathbf{z} \to$ image

12.10 Classifier-Free Guidance

12.10.1 The Problem

- How strongly to follow text prompt?
- Need control: adherence vs diversity

12.10.2 The Solution

- Train one model with AND without conditioning
- During training: randomly drop text (e.g., 10% of time)
- During generation:
  - $\boldsymbol{\epsilon}_{\text{guided}} = \boldsymbol{\epsilon}_{\text{uncond}} + s \cdot (\boldsymbol{\epsilon}_{\text{cond}} - \boldsymbol{\epsilon}_{\text{uncond}})$
  - s = guidance scale (typically 7-15)

12.10.3 Why It Works

- Unconditional: base distribution
- Conditional: distribution given text
- Difference: direction toward text alignment
- Amplify: move further in that direction
- Trade: prompt adherence vs diversity

12.10.4 Current Standard

- No separate classifier needed
- Simple training modification
- Effective control mechanism
- Now ubiquitous in text-to-image

12.11 Multimodal Models: Foundational References

- **Cross-Modal Attention Mechanism**
  - "Show, Attend and Tell" Xu et al. [2015]
  - **Breakthrough:** *Cross-modal attention* – queries from one modality attend to keys/values from another
  - Enabled fine-grained, spatially-aware conditioning across modalities
  - Foundation for attention-based multimodal generation
- **CLIP (Contrastive Language-Image Pre-training)**
  - Radford et al. [2021] Radford et al. [2021]
  - **Breakthrough:** *Contrastive learning for vision-language alignment*
  - Shared semantic embedding space for images and text
  - Enabled zero-shot classification and cross-modal retrieval
  - Foundation for text-conditioned image generation
- **DALL-E**
  - Ramukumar et al. [2021] Kumar and Levine [2021]
  - **Breakthrough:** *VQ-VAE + autoregressive Transformer* for text-to-image generation
  - Discrete visual codebook + AR modeling over codes conditioned on text
  - First large-scale text-to-image generation with Transformers
- **DALL-E 2**
  - Ramesh et al. [2022] Ramesh et al. [2022]
  - **Breakthrough:** *CLIP embeddings + diffusion* for text-to-image
  - Prior network: text $\rightarrow$ CLIP image embedding $\rightarrow$ diffusion decoder
  - Higher quality and resolution than DALL-E
- **Latent Diffusion Models / Stable Diffusion**
  - Rombach et al. [2022] (for multimodal aspect)
  - **Claimed Breakthrough:** *Cross-modal conditioning in latent diffusion space*
  - Text conditioning via cross-attention to CLIP text embeddings
  - Efficient high-resolution text-to-image generation
- **Imagen**
  - Saharia et al. [2022] Saharia et al. [2022]
  - **Breakthrough:** *Large frozen language model (T5) as text encoder*
  - Deep semantic text understanding via pre-trained LLM
  - Cascaded diffusion for progressive upsampling
  - Showed importance of strong language understanding for text-to-image

12.12 Task Performance Analysis

MODELING ✓✓

- Flexible: cross-modal conditioning
- Requires: alignment + conditioning mechanism
- Quality depends on both components

LEARNING ✓

- Stable when built on stable base models
- Contrastive alignment scales well
- Need paired data (web-scale available)

SCORING ∼

- Depends on underlying model (Diffusion: approximate)
- CLIP provides cross-modal similarity scores

SAMPLING

- Speed from base model (Diffusion: slower)
- Guidance adds cost (2× passes for classifier-free)
- Quality excellent with proper guidance

INFERENCE

- Text encoder: fast
- Cross-modal retrieval possible (CLIP)
- Conditioning strength controllable

12.13 Fundamental Tradeoff

- **Wins**: Cross-modal generation, semantic control, web-scale learning
- **Sacrifices**: Need paired data, guidance overhead, depends on alignment quality
- **When**: Cross-modal applications, especially text-to-image

12.14 Why This Matters: Extension via Composition

- Not new generative model type
- **Combination** of existing pieces:
  - Generative model (Diffusion)
  - Alignment mechanism (CLIP)
  - Conditioning architecture (cross-attention)
  - Control mechanism (classifier-free guidance)
- Models extend naturally to new settings
- Same principles applied in new context

12.15 The Schema

- Single modality: AR, VAE, Diffusion, etc.
- Composition: Latent Diffusion (speed problem)
- Extension: Multimodal (cross-modal generation)
- Each step: natural question $\rightarrow$ solution
- Design space is structured, not ad hoc

Part III

Note on Evaluation

13 Evaluation of Generative Tasks

13.1 Why Fundamentally Different and Harder

Evaluating generative models presents challenges fundamentally distinct from discriminative tasks.

Many-to-one vs. one-to-many:

- **Discriminative:** Many inputs → one label. Clear ground truth. "Is this a cat?" has a definite answer.
- **Generative:** One description → many valid outputs. "Generate a cat" has infinitely many valid answers.

No unique correct answer: Unlike classification accuracy or regression error, there is no single ground truth sample. Multiple samples can be equally valid or invalid in different ways.

Cannot directly compare distributions: We want $p_{\text{model}} \approx p_{\text{data}}$, but we cannot compute this directly. We only have:

- Samples from p_{data} (the training/test data)
- Samples from p_{model} (generated outputs)
- Sometimes: $p_{\text{model}}(x)$ for given x (but never $p_{\text{data}}(x)$)

All evaluation metrics are indirect proxies for the distribution match we truly care about.

13.2 The Fundamental Trade-off: Two Directions of KL Divergence

The central challenge: we cannot minimize both directions of divergence simultaneously (unless $p_{\text{model}} = p_{\text{data}}$).

Forward KL divergence:

$$D_{KL}(p_{\text{data}} \| p_{\text{model}}) = \mathbb{E}_{x \sim p_{\text{data}}} \left[\log \frac{p_{\text{data}}(x)}{p_{\text{model}}(x)} \right] \tag{83}$$

- **Measures:** When data is likely, is the model also likely there?
- **Penalizes:** Missing modes or regions where data exists
- **Encourages:** Coverage – don't miss any part of the distribution
- **Risk:** Spreading probability mass into unrealistic regions to avoid missing anything
- **This is RECALL/COVERAGE**

Reverse KL divergence:

$$D_{KL}(p_{\text{model}} \| p_{\text{data}}) = \mathbb{E}_{x \sim p_{\text{model}}} \left[\log \frac{p_{\text{model}}(x)}{p_{\text{data}}(x)} \right] \tag{84}$$

- **Measures:** When the model is likely, is data also likely there?
- **Penalizes:** Generating samples where data doesn't exist
- **Encourages:** Precision – don't generate unrealistic samples
- **Risk:** Mode collapse (safe to focus on one mode and ignore others)

- **This is PRECISION**

The unavoidable trade-off: Perfect coverage risks generating unrealistic samples. Perfect precision risks missing parts of the distribution. Different evaluation metrics implicitly privilege one direction over the other.

Other tensions:

- **Likelihood vs. perception:** High likelihood does not guarantee perceptually good samples. Models can achieve high likelihood by modeling imperceptible high-frequency details while missing perceptually important structure.
- **Automatic vs. human evaluation:** Automatic metrics scale but may not align with human judgment. Human evaluation provides ground truth but is expensive, slow, and subjective.

13.3 The Benchmark Crisis

13.3.1 The Essential Role of Benchmarks

Rigorous evaluation requires:

- **Standard, held-out test sets:** Data the model has never seen during training
- **Community agreement:** Shared protocols for reproducibility and fair comparison
- **Evaluation discipline:** Preventing selective reporting and ensuring honest assessment

The cherry-picking crisis: Without standard benchmarks and evaluation protocols, selective reporting undermines all published results:

- Choosing favorable benchmarks while ignoring unfavorable ones
- Reporting best runs while hiding failed attempts
- Showing curated examples while hiding typical outputs
- Comparing against weak or outdated baselines
- Making claims irreproducible and unreliable

13.3.2 The Crisis: Opaque Contamination at Scale

Modern generative AI faces a crisis from the deadly compounding of three factors:

1. Contamination: Models may have seen test data during training

- *Alone:* Manageable if you know what was trained on
- Can verify and create clean benchmarks

2. Scale: Training data is "all documented knowledge"

- *Alone:* Hard to find held-out data, but at least composition is knowable
- Can attempt to create novel test data

3. Opacity: Training data composition is unpublished (closed models)

- *Alone:* Frustrating but could rely on known-clean benchmarks
- Requires trust in model providers

The Crisis: When all three factors combine:

- You cannot verify what was trained on (opacity)
- Training likely included everything publicly available (scale)
- Including most or all existing test sets (contamination)
- Making ALL traditional evaluation metrics potentially meaningless
- No way to distinguish memorization from generalization
- Evaluation becomes trust-based rather than verification-based

13.3.3 Attempted Solutions and Their Limitations

Dynamic benchmarks: Generate test cases on-the-fly

- *Limitation:* Expensive, hard to standardize, can still be included in next training run

Human evaluation: Direct assessment by humans

- *Limitation:* Doesn't scale, expensive, subjective, can still be gamed through cherry-picking

Adversarial test construction: Deliberately create hard cases

- *Limitation:* Arms race, published adversarial examples become training data

Time-stamped test sets: Use data created after training cutoff

- *Limitation:* Only works if training cutoff is verifiable (requires transparency)

None of these solutions fully address the contamination-at-opaque-scale problem. The field currently lacks a comprehensive answer to this evaluation crisis.

13.4 Evaluation Methods

We now survey specific metrics, understanding that each represents an imperfect proxy for the distribution match we truly seek.

13.4.1 Likelihood-Based Metrics

When we can compute $p_{\text{model}}(x)$, we can directly measure how well the model assigns probability to real data.

Negative Log-Likelihood (NLL):

$$\mathcal{L}_{\text{NLL}} = -\mathbb{E}_{x \sim p_{\text{data}}}[\log p_{\text{model}}(x)] \tag{85}$$

Perplexity (for sequences):

$$\text{Perplexity} = \exp(\mathcal{L}_{\text{NLL}}) \tag{86}$$

ELBO (for VAEs):

$$\mathcal{L}_{\text{ELBO}} = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x) \| p(z)) \tag{87}$$

Lower bound on log-likelihood; the actual likelihood is intractable.

Available for: Autoregressive models (tractable by chain rule), normalizing flows (tractable via change of variables), VAEs (via ELBO lower bound)

Not available for: GANs (implicit density), diffusion models (intractable, though boundable)

Why the limitation: Likelihood-based metrics implicitly optimize forward KL, encouraging coverage. This can lead to:

- Wasting probability mass on imperceptible high-frequency details
- Spreading mass over blurry averages to avoid missing modes
- High likelihood not correlating with perceptual quality

The likelihood-perception disconnect is fundamental: likelihood treats all dimensions equally, but human perception is highly non-uniform.

13.4.2 Sample-Based Metrics

When likelihood is unavailable or unreliable, we evaluate the samples themselves.

Inception Score (IS):

$$\text{IS} = \exp\left(\mathbb{E}_{x\sim p_{\text{model}}}[D_{KL}(p(y|x)\|p(y))]\right) \tag{88}$$

where $p(y|x)$ is from a pre-trained classifier (Inception network), $p(y) = \mathbb{E}_x[p(y|x)]$.

Core idea: Good samples should be recognizable as specific classes (low entropy per sample) AND diverse across samples (high marginal entropy).

Why the limitation:

- Conflates precision and recall into a single number
- Biased by pre-trained classifier (typically ImageNet)
- Insensitive to mode dropping within classes
- Can be fooled by generating one perfect sample per class

Fréchet Inception Distance (FID):

$$\text{FID} = \|\mu_r - \mu_g\|^2 + \text{Tr}\left(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}\right) \tag{89}$$

where (μ_r, Σ_r) and (μ_g, Σ_g) are mean and covariance of real and generated samples in Inception feature space.

Core idea: Compare the distribution of generated samples to real samples in a learned feature space.

Why the limitation:

- Still depends on feature extractor choice (Inception network bias)
- Doesn't explicitly separate precision from recall
- Requires many samples for stable covariance estimates
- Sensitive to implementation details

Better correlation with human judgment than IS, but still imperfect.

Precision and Recall for Distributions:

Core idea: Explicitly separate precision (are generated samples realistic?) from recall (does the model cover the data distribution?).

Various formulations exist; the intuition:

- **Precision:** Fraction of generated samples that lie near the real data manifold
- **Recall:** Fraction of the real data manifold covered by generated samples

Why the limitation:

- Depends on choice of feature space and manifold definition
- Requires manifold estimation (computationally expensive, potentially unstable)
- Still uses features from pre-trained networks

Critical note: Real samples must be held-out test data, NOT training data. Using training data rewards memorization rather than generalization. Let M = number of generated samples, N = number of held-out test samples.

13.4.3 Human Evaluation

Core idea: Direct assessment by humans, typically through preference judgments or quality ratings.

Mean Opinion Score (MOS):

$$\text{MOS} = \frac{1}{N \times M} \sum_{i=1}^{N} \sum_{j=1}^{M} r_{ij} \tag{90}$$

where r_{ij} is rating by human j for sample i.

Why it's important:

- Only true measure of perceptual quality
- Ground truth for what we ultimately care about
- Can capture nuanced qualities automatic metrics miss

Why the limitation:

- Doesn't scale – expensive and slow
- Subjective – inter-rater agreement varies
- Inconsistent – same rater may judge differently over time
- Can still be manipulated through cherry-picking which samples to evaluate

13.4.4 Task-Specific Metrics

For conditional generation tasks (text→image, image→text, text→text), we can measure alignment between condition and output.

Examples:

- **CLIP Score** (text-image alignment): Cosine similarity in CLIP embedding space
- **BLEU** (text generation): N-gram overlap with reference translations
- **Accuracy on downstream tasks:** For representation learning (e.g., image classification using learned features)

Why useful: Conditional tasks have clearer evaluation criteria than unconditional generation.

Why limited: Still proxy metrics – high CLIP score doesn't guarantee perceptually good images; high BLEU doesn't guarantee good translations.

13.4.5 LLM-Based Evaluation

Core idea: Use large language models (GPT-4, Claude, etc.) as judges to evaluate qualities like coherence, instruction-following, helpfulness, factuality.

Absolute scoring:

$$\text{Score} = \text{LLM}(\text{prompt}, x_{\text{generated}}, \text{criteria}) \tag{91}$$

Pairwise comparison:

$$P(x_A \succ x_B) = \text{LLM}(\text{prompt}, x_A, x_B, \text{criteria}) \tag{92}$$

Why it's useful:

- Scalable compared to human evaluation (though still expensive)
- Can capture nuanced qualities hard to formalize (coherence, helpfulness, instruction-following)
- Reasonable correlation with human judgment for many tasks
- Provides detailed critiques, not just scores

Why the limitations:

- Inherits biases of the judge LLM (position bias, verbosity bias, self-preference)
- Not truly ground truth – just another model's opinion
- Can be gamed or exploited by optimizing for judge preferences
- Correlation with human judgment varies significantly by task and domain
- Still expensive compared to fully automatic metrics
- Evaluation model may have been trained on outputs being evaluated (contamination at meta-level)

LLM-based evaluation sits between automatic metrics (scalable but potentially misaligned) and human evaluation (ground truth but expensive).

Part IV

Technical Horizons: Beyond this Tutorial

14 What We Did Not Cover

This tutorial focused on foundational generative models for images and sequences. Many important topics remain beyond our scope:

14.1 Energy-Based Models

- **What:** Model unnormalized probability $p(x) \propto \exp(-E(x))$ where $E(x)$ is a learned energy function
- **Challenge:** No tractable likelihood (partition function intractable), requires special sampling and training procedures
- **Different paradigm:** Neither explicit density (like flows/AR) nor implicit (like GANs), but energy-based
- **Why important:** Unified framework connecting generative and discriminative models, flexible modeling
- **Foundational references:** [Du & Mordatch 2019], [Song & Kingma 2021]

14.2 Video and Temporal Generation

- **What:** Extending generative models to temporal sequences of high-dimensional data (video, long time series)
- **Challenge:** Modeling both spatial structure (within frames) and temporal coherence (across frames)
- **Scales beyond sequences:** While autoregressive models handle 1D sequences, video requires modeling 3D structure (height $\times$ width $\times$ time)
- **Why important:** Central to video synthesis, animation, temporal prediction
- **Foundational references:** [Ho et al. 2022 - Video Diffusion], [OpenAI 2024 - Sora]

14.3 3D Generation

- **What:** Generating three-dimensional shapes, scenes, and objects with geometric and appearance consistency
- **Representations:** Voxels, point clouds, meshes, implicit functions (NeRF), 3D-aware GANs
- **Challenge:** Modeling 3D structure from 2D supervision, ensuring multi-view consistency
- **Connection to covered material:** Often combines diffusion or GANs with 3D representations
- **Why important:** Computer graphics, AR/VR, robotics, scientific visualization
- **Foundational references:** [Mildenhall et al. 2020 - NeRF], [Poole et al. 2022 - DreamFusion]

14.4 Graph and Structured Generation

[See Appendix]

- **What:** Generating discrete structured data: graphs, trees, syntax structures, molecular structures
- **Challenge:** Modeling complex dependencies and constraints (connectivity, validity, symmetries)
- **Discrete and structured:** Unlike images (continuous, grid) or text (1D sequence), graphs have arbitrary topology
- **Applications:** Molecular design, program synthesis, knowledge graphs, social networks
- **Why important:** Many real-world problems involve structured discrete data
- **Foundational references:** [You et al. 2018 - GraphRNN], [Jin et al. 2018 - Junction Tree VAE]

14.5 Efficient Training and Inference

- **What:** Methods to reduce computational cost of training and sampling from generative models
- **Training efficiency:** Mixed precision, efficient architectures, better optimization
- **Inference efficiency:** Distillation (fewer sampling steps), quantization, caching, parallel decoding
- **Why critical:** Diffusion models require hundreds of steps; autoregressive models are sequential; both are expensive
- **Trade-offs:** Speed vs. quality, model size vs. capability
- **Foundational references:** [Salimans & Ho 2022 - Progressive Distillation], [Song et al. 2023 - Consistency Models]

14.6 Interpretability and Mechanistic Understanding

- **What:** Understanding what generative models learn, how they represent knowledge, and why they make specific outputs
- **Questions:** What features do different layers encode? How do models compose concepts? Where is knowledge stored?
- **Methods:** Activation analysis, causal interventions, circuit discovery, feature visualization
- **Why important:**
  - Understanding failure modes and biases
  - Improving model design and training
  - Safety and alignment (especially for LLMs)
  - Scientific understanding of representation learning
- **Challenge:** Models are increasingly large and complex, making interpretation harder as capabilities grow

- **Foundational references:** [Olah et al. 2020 - Circuits Thread], [Elhage et al. 2021 - Mathematical Framework]

These topics represent active research frontiers. Many build on the foundations we covered—applying VAEs, GANs, flows, or diffusion to new domains or improving their efficiency. Others, like energy-based models and interpretability, ask fundamentally different questions about generative modeling.

15 The Frontiers of Generative Modeling

We now turn to the frontiers – the open challenges that define the future of generative AI. We organize these into two parts: the capabilities we seek (Part 1) and the technical approaches and understanding we pursue to achieve them (Part 2).

15.1 Part 1: Task, Application, and Utility Frontiers

These are the capabilities we want generative models to achieve – the *what* of future AI systems.

15.1.1 World Models and System 1+2 Integration

- Building internal models of causality, physics, dynamics – how the world actually works
- **System 1:** Fast, intuitive, pattern-matching reasoning (closer to current LLMs)
- **System 2:** Slow, deliberate reasoning through world models – explicit simulation, counterfactuals, causal reasoning
- **Integration:** Knowing when to use pattern matching vs model-based reasoning
- Not just statistical patterns, but understanding – the difference between correlation and causation
- System 2 is *determined by* world models – you cannot do explicit reasoning without internal models of how things work

15.1.2 Efficient Learning of Knowledge Representation

- **Sample efficiency:** Learning from far fewer examples (humans learn multiplication from $\sim$100 examples, not billions)
- **Parametric efficiency:** Storing knowledge as algorithms, not lookup tables
- Don't memorize what can be computed – store the multiplication algorithm, not the multiplication table
- Learning the right level of abstraction from data
- These two are unified: if you learn the *computable structure* (algorithm), you need both fewer examples AND less storage
- Current models closer to memorization than abstraction learning

15.1.3 Contextual Memorization vs Creative Hallucination

- Recognizing when factual precision is required (math exam, medical diagnosis) vs when creativity and imagination are essential (poetry, art, fiction)
- **Not about approximation:** This is about mode-switching between factual and imaginative generation
- Creative hallucination is not a bug – it gave us Carroll, Lear, Dalí, Picasso
- But you cannot hallucinate in a math exam
- The challenge: context-aware recognition of which mode is appropriate
- "To hallucinate or not to hallucinate" – context determines appropriateness

15.1.4 Task Decomposition and Solution Recomposition

- Breaking complex problems into manageable sub-problems
- Solving sub-problems independently (potentially in parallel, with different tools/methods)
- Reintegrating partial solutions into coherent whole
- Hierarchical and compositional problem-solving
- Requires understanding task structure and dependencies
- Challenge: maintaining consistency and coherence across decomposition and recomposition

15.2 Part 2: Technical Frontiers and Fundamental Understanding

These are the methods being developed (the *how*) and the insights being sought (the *why*) to address the task frontiers and understand what we have built.

15.2.1 Neurosymbolic Techniques

Addresses: Task 1 – World Models and System 1+2 Integration

- **Techniques:** Combining neural networks (pattern matching, continuous) with symbolic reasoning (logic, discrete)
- Explicit causal models integrated with learned representations
- Hybrid architectures that leverage both paradigms
- **Understanding sought:** How to integrate pattern recognition with logical reasoning? What representations enable both?
- **Challenge:** Bridging continuous (neural) and discrete (symbolic) representations, learning when to invoke which mode

15.2.2 Meta-Learning and Few-Shot Learning

Addresses: Task 2 – Efficient Learning of Knowledge Representation

- **Techniques:** Learning to learn, curriculum learning, transfer learning, in-context learning
- Discovering abstractions that generalize across tasks
- Learning from demonstrations and analogies
- **Understanding sought:** What makes good abstractions? What is the right inductive bias? What enables generalization from few examples?
- Sample complexity theory – fundamental limits on learning from limited data
- **Challenge:** Moving from memorization to true abstraction learning

15.2.3 Contextual Mode Recognition

Addresses: Task 3 – Contextual Memorization vs Creative Hallucination

- **Current state:** Open problem – no explicit techniques currently addressing this
- **Understanding needed:** How do models recognize task context? What signals in the input indicate factual vs creative mode is appropriate?
- **Related work:** Instruction following, prompt engineering address this superficially but don't solve the core issue
- **Challenge:** Defining what contexts demand precision vs creativity, detecting these contexts, switching modes appropriately
- This may require models to understand human values and social context

15.2.4 Agentic Systems and Tool Use

Addresses: Task 4 – Task Decomposition and Solution Recomposition

- **Techniques:** Multi-agent systems, tool and API integration, chain-of-thought prompting, hierarchical planning
- ReAct (reasoning + acting), allowing models to interact with external tools and environments
- Breaking tasks into subtasks, delegating to specialized modules or external tools
- **Understanding sought:** How to decompose tasks effectively? When to use which tools? How to recompose solutions while maintaining coherence?
- **Challenge:** Credit assignment across decomposed tasks, managing dependencies, verifying correctness of subcomponents

15.2.5 Fundamental Understanding

Meta-Frontier: Informs all tasks

This frontier is about understanding *why* current approaches work, what they learn, and what their fundamental limits are. Progress here could inform all other frontiers.

Emergence and Scaling Laws:

- Why do new capabilities emerge as models scale?
- Are emergent capabilities predictable or surprising?
- Phase transitions in model behavior with scale
- What are the limits of scaling? Do capabilities saturate?
- Understanding emergence could reveal what's fundamentally achievable

Mechanistic Interpretability:

- What do models learn internally? How do they represent knowledge?
- Circuit discovery – identifying minimal subgraphs that implement specific capabilities
- Feature visualization and activation analysis

- Causal interventions – what happens when we modify internal representations?
- Understanding the *mechanism* could enable more principled design

Theoretical Foundations:

- Why do these architectures work in principle?
- Generalization bounds for deep learning and generative models
- What are fundamental computational and information-theoretic limits?
- Mathematical frameworks connecting architecture, training, and capabilities
- Theory could predict what's possible before building it

Part V
AI and the World

16 Historical Foundations: The Debts We Inherit

16.1 Introduction: Standing on Centuries of Shoulders

We have now explored modern generative AI:

- The fundamental tasks
- Operations and frameworks
- Major model categories
- Frontier directions: System II reasoning, neuro-symbolic AI, world models, causal understanding

None of this emerged in a vacuum. Every technique rests on centuries of mathematical, scientific, and philosophical foundations spanning cultures, continents, and eras.

This section acknowledges two kinds of debt:

1. **Recent foundations (Modern AI History)**: Documented but at risk of being forgotten
   - AI pioneers from 1940s-2000s whose work we know but undervalue
   - Reminder: Modern AI didn't start in 2012, 2017, or 2020
2. **Deep foundations (by Task)**: Never even been acknowledged
   - Mathematical and scientific developments from antiquity onward
   - Most AI practitioners don't know these connections exist
   - Goes back millennia, across all cultures

Why this matters:

- **Humility**: We're custodians of accumulated knowledge, not creators ex nihilo
- **Global foundations**: Critical contributions from every continent, culture, era
- **Connections**: Ancient insights enable modern breakthroughs in surprising ways
- **Responsibility**: As inheritors, what do we owe in return?

16.2 Modern AI History: Recent Foundations at Risk of Being Forgotten

16.2.1 Birth of Artificial Intelligence (1940s-1960s)

- Computer architecture, early neural concepts [von Neumann, 1903-1957, Hungary/US]
- Information retrieval vision [Bush, 1890-1974, US]
- Coined "artificial intelligence", LISP [McCarthy, 1927-2011, US]
- MIT AI Lab, neural networks, symbolic AI [Minsky, 1927-2016, US]
- Logic Theorist, General Problem Solver [Newell/Simon, 1927-1992/1916-2001, US]
- Machine learning pioneer, MENACE [Michie, 1923-2007, Britain]
- Formal definition of ML [Mitchell, 1951-, US]

16.2.2 Modern Machine Learning Era (1980s-2020s)

- Probabilistic graphical models [Jordan, 1956-, US]
- Backpropagation, Boltzmann machines, deep learning [Hinton, 1947-, Britain/Canada]
- Rediscovered backpropagation, PDP framework [Rumelhart, 1942-2011, US]

These pioneers are documented. But even documented history risks being forgotten.

Now we go deeper—to foundations that were never acknowledged at all.

16.3 TASK 1: Modeling

Fundamental Tradeoffs:

- Expressiveness vs tractability
- Discrete vs continuous representations
- Parametric vs non-parametric
- Structure vs capacity

16.3.1 Probability Theory

Role in Modeling: Mathematical framework for representing uncertainty; defines distributions over data; measures likelihood; foundation for all generative models.

- Early cryptanalysis, frequency analysis [Al-Kindi, 801-873, Baghdad]
- Systematic probability [Cardano, 1501-1576, Italy]
- Probability foundations [Pascal/Fermat, 17th C, France]
- Law of large numbers [Jacob Bernoulli, 1655-1705, Switzerland]
- Normal distribution, central limit theorem [de Moivre, 1667-1754, France/Britain]
- Bayes' theorem [Bayes, 1701-1761, Britain]
- Frequency interpretation [Venn, 1834-1923, Britain]
- Measure-theoretic foundations [Kolmogorov, 1933, Russia]

16.3.2 Markov Processes

Role in Modeling: Sequential dependencies; temporal structure; autoregressive generation; state transitions; foundational for sequence models and diffusion.

- Markov property, Markov chains [Markov, 1913, Russia]
- Stochastic process theory [Doob, 1953, US]
- Stochastic calculus [Itô, 1944, Japan]

16.3.3 Linear Algebra & Dimensionality Reduction

Role in Modeling: Representation as transformation; decomposing structure into components; dimensionality reduction; learned embeddings as nonlinear generalizations. **Essential for encoding/representation learning**—VAE encoders are nonlinear descendants of PCA.

- Matrix methods for linear systems [Nine Chapters, 200 BCE, China]
- Cartesian coordinate system [Descartes, 1596-1650, France]
- Gaussian elimination [Gauss, 1777-1855, Germany]
- Matrix algebra, eigenvalues [Sylvester, 1814-1897 & Cayley, 1821-1895, Britain]
- Orthogonal decompositions, SVD [Schmidt, 1907, Germany]
- Principal component analysis [Hotelling, 1933, US]
- Factor analysis [Harman, 1960s & Jöreskog, 1970s, US/Sweden]

16.3.4 Graph Theory

Role in Modeling: Represents relational structure; encodes conditional independence in PGMs; molecular graphs; knowledge graphs.

- Founded graph theory, Seven Bridges [Euler, 1736, Switzerland]
- Random graph models [Erdős/Rényi, 1959-60, Hungary]
- Scale-free networks, preferential attachment [Barabási/Albert, 1999, Hungary-US/Romania]
- Bayesian networks [Pearl, 1988, Israel/US]
- Junction tree algorithm [Lauritzen/Spiegelhalter, 1988, Denmark/Britain]
- Comprehensive graphical models [Koller/Friedman, 2009, Israel/US]

16.3.5 Statistical Mechanics & Physical Chemistry

Role in Modeling: Energy-based models; partition functions; Boltzmann distributions; free energy principles underlying variational inference.

- Boltzmann distribution, entropy [Boltzmann, 1844-1906, Austria]
- Ensemble theory, Gibbs distribution [Gibbs, 1839-1903, US]
- Helmholtz free energy [von Helmholtz, 1882, Germany]
- Ising model [Ising, 1925, Germany]
- Mean field approximation [Weiss, 1907, France]
- Free energy to variational principles [Feynman, 1972, US]
- Free energy for neural computation [Friston, 2010, Britain]

16.3.6 Logic & Formal Systems

Role in Modeling: Formal reasoning; knowledge representation; structured rules. **Essential for System II reasoning and neuro-symbolic AI** (see Frontiers).

- Logic and epistemology [Nyaya Sutra, 6th-5th C BCE, India]
- Syllogistic logic [Aristotle, 384-322 BCE, Greece]
- Propositional logic [Ibn Sina/Avicenna, 980-1037, Persia]
- Predicate logic [Frege, 1879, Germany]
- Principia Mathematica [Russell/Whitehead, early 20th C, Britain]
- Incompleteness theorems [Gödel, 1906-1978, Austria/US]

16.3.7 Linguistics & Formal Grammar

Role in Modeling: Structure of language; compositional semantics; formal language hierarchy. **Foundational for understanding sequential structure** in language models.

- Formal grammar [Panini, 6th-4th C BCE, India]
- Structural linguistics [de Saussure, 1857-1913, Switzerland]
- Chomsky hierarchy, generative grammar [Chomsky, 1928-, US]
- Distributional hypothesis [Harris, 1954, US]
- Vector space semantics [Turney/Pantel, 2010, Canada/US]

16.3.8 Neural Computation as Modeling Framework

Role in Modeling: Biological neurons as computational units; networks of simple units as universal computing framework; inspiration for artificial neural network architecture.

- Neurons as logical units [McCulloch/Pitts, 1943, US]
- Perceptron [Rosenblatt, 1958, US]
- Hopfield networks [Hopfield, 1982, US]

16.4 TASK 2: Learning

Fundamental Tradeoffs:

- Bias-variance tradeoff
- Sample complexity vs model capacity
- Exploration vs exploitation
- Gradient-based vs gradient-free optimization

16.4.1 Calculus & Differentiation

Role in Learning: Enables gradient-based optimization; chain rule enables backpropagation; analyzing convergence.

- Method of exhaustion [Archimedes, 287-212 BCE, Greece]
- Calculus foundations [Newton, 1643-1727 & Leibniz, 1646-1716, Britain/Germany]
- Chain rule formulation [Leibniz, 1676, Germany]
- Chain rule development [Johann Bernoulli, 1690s-1710s, Switzerland]
- Rigorous calculus, chain rule proof [Cauchy, 1789-1857, France]

16.4.2 Least Squares & Regression

Role in Learning: Prototype of learning from data; fitting models to observations; foundational loss function.

- Method of least squares [Gauss, 1777-1855 & Legendre, 1752-1833, Germany/France]
- Correlation and regression [Galton, 1822-1911, Britain]
- Maximum likelihood estimation [Fisher, 1890-1962, Britain]

16.4.3 Optimization Theory

Role in Learning: Training objectives; finding optimal parameters; gradient descent; convergence guarantees.

- Optimality conditions [Fermat, 1607-1665, France]
- Lagrange multipliers [Lagrange, 1788, Italy/France]
- Gradient descent [Cauchy, 1847, France]
- Stochastic gradient descent [Robbins/Monro, 1951, US]
- Momentum methods [Polyak, 1964, Soviet Union/Russia]
- Accelerated gradient methods [Nesterov, 1983, Soviet Union/Belgium]
- Adam optimizer [Kingma/Ba, 2015, Netherlands/Canada]

16.4.4 Optimal Transport

Role in Learning: Alternative to KL divergence for comparing distributions; Wasserstein distance; optimal mappings. **Used in Wasserstein GANs** for stable training.

- Optimal transport problem [Monge, 1746-1818, France]
- OT as linear programming [Kantorovich, 1912-1986, Soviet Union/Russia]
- Wasserstein distance [Vaserstein, 1940-, Russia/US]
- Modern OT theory [Villani, 2009, France]

16.4.5 Neural Learning Rules

Role in Learning: Biological inspiration for learning; Hebbian learning; associative memory; backpropagation algorithm.

- Hebbian learning [Hebb, 1949, Canada]
- Backpropagation [Rumelhart/Hinton, 1986, US/Canada]
- Neural network models of cognition [Hinton, 1947-, Britain/Canada]

16.4.6 Dynamic Programming & Control

Role in Learning: Primary: Exact inference in PGMs via recursive decomposition; forward-backward algorithm; variable elimination. **Secondary:** Reinforcement learning foundations; value functions; policy iteration.

- Dynamic programming [Bellman, 1950s, US]
- Optimal control, maximum principle [Pontryagin, mid-20th C, Soviet Union]

16.4.7 Sample Complexity Theory

Role in Learning: How much data is needed for generalization; PAC learning; VC dimension.

- VC theory, sample complexity [Vapnik/Chervonenkis, 1971, Soviet Union/Russia]

16.5 TASK 3: Scoring

Fundamental Tradeoffs:

- Exact vs approximate likelihood
- Tractable computation vs model expressiveness
- Per-sample scoring vs amortized inference

16.5.1 Information Theory

Role in Scoring: Measuring uncertainty; cross-entropy loss; KL divergence; mutual information.

- Information theory foundations, entropy [Shannon, 1948, US]
- KL divergence [Kullback/Leibler, 1951, US]
- Rate-distortion theory [Shannon, 1959, US]

16.6 TASK 4: Sampling

Fundamental Tradeoffs:

- Exact vs approximate sampling
- Sample quality vs computational cost
- Independent vs correlated samples

16.6.1 Monte Carlo Methods

Role in Sampling: Sampling from complex distributions; approximating intractable integrals; MCMC for inference.

- Monte Carlo methods [Ulam/Metropolis, 1909-1984/1915-1999, Poland-US/Greece-US]
- Metropolis algorithm [Metropolis et al., 1953, US]
- Metropolis-Hastings [Hastings, 1970, Canada]

16.6.2 Langevin Dynamics & Score-Based Methods

Role in Sampling: Continuous-time sampling; gradient-based generation. **Foundation for diffusion models.**

- Stochastic differential equations [Itô, 1944, Japan]

16.7 TASK 5: Inference

Fundamental Tradeoffs:

- Exact vs approximate inference
- Computational cost vs accuracy
- Variational vs sampling-based inference

16.7.1 Bayesian Inference

Role in Inference: Posterior computation; belief updating; probabilistic reasoning; latent variable inference.

- Bayes' theorem [Bayes, 1701-1761, Britain]
- Bayesian inference development [Laplace, 1749-1827, France]
- Revival of Bayesian methods [Jeffreys, 1891-1989, Britain]

16.7.2 Statistical Inference

Role in Inference: Estimating parameters; confidence intervals; hypothesis testing.

- Central limit theorem [de Moivre, 1667-1754 & Laplace, 1749-1827, France/Britain]
- Law of large numbers [Jacob Bernoulli, 1655-1705, Switzerland]
- Maximum likelihood estimation [Fisher, 1890-1962, Britain]

16.7.3 Variational Methods & Free Energy

Role in Inference: Approximate Bayesian inference; ELBO optimization; **VAE training**; mean field approximation.

- Free energy minimization [von Helmholtz, 1882, Germany]
- Variational methods in quantum mechanics [Feynman/Hibbs, 1965, US]
- Mean field approximation [Weiss, 1907, France]
- Free energy principle [Friston, 2010, Britain]

16.7.4 Message Passing & Belief Propagation

Role in Inference: Exact inference on tree-structured PGMs; iterative algorithms; distributed computation.

- Belief propagation [Gallager, 1962, US]
- Junction tree algorithm [Lauritzen/Spiegelhalter, 1988, Denmark/Britain]

16.7.5 Kalman Filtering & State Estimation

Role in Inference: Sequential Bayesian inference; recursive estimation; optimal filtering for linear-Gaussian systems.

- Kalman filtering [Kalman, 1960, Hungary/US]

16.7.6 Causal Inference

Role in Inference: Identifying causal structure; interventional distributions; counterfactual reasoning. **Essential for System II reasoning and world models** (see Frontiers).

- Causal DAGs, do-calculus [Pearl, 1988/2009, Israel/US]
- Potential outcomes [Rubin, 1974, US]

16.7.7 Signal Processing & Filtering

Role in Inference: Separating signal from noise; optimal filtering; recursive estimation.

- Cybernetics, Wiener filter [Wiener, 1930s-1949, US]

16.8 Reflection: What We Owe

Patterns that emerge:

No Single Origin Story

- Contributions from every inhabited continent
- Multiple independent discoveries (calculus, probability)
- Cross-cultural transmission essential
- Modern AI is profoundly multicultural

Time Scales of Progress

- Centuries from theory to application
- Sudden breakthroughs rest on decades of "useless" pure research
- Cutting-edge today has roots in 1700s, sometimes BCE

Persistent Tradeoffs

- Structure vs capacity (1700s factorization vs 2010s neural nets)
- Exact vs approximate (always a tension)
- Computational cost vs quality (from Gauss to today)

Openness and Sharing

- Progress accelerated by publication, not secrecy
- Mathematical knowledge as public good
- Standing on shoulders requires willingness to share

16.9 Forward: From Debt to Responsibility

Understanding these debts shapes our ethical obligations:

- We benefit from **centuries of shared knowledge**—what do we owe in return?
- Ancient mathematicians couldn't foresee modern AI—**what unforeseeable consequences** might our work have?
- This technology draws on **global intellectual traditions**—who benefits from its deployment?
- Progress came from **open sharing and collaboration**—how do we honor that tradition?

These questions transition to the next section on Ethics and Societal Impact, where we examine our responsibilities as inheritors and developers of this powerful technology.

The debts we've inherited are not just technical—they're **moral obligations** to use this knowledge wisely, share it generously, and consider its impacts thoughtfully.

17 AI Benefits

17.1 Documented Present Benefits (Here and Now)

Healthcare & Medicine

- Medical imaging analysis and diagnosis assistance
- Drug candidate identification and screening
- Protein structure prediction [AlphaFold]
- Administrative burden reduction (documentation, scheduling)
- Radiology and pathology support

Scientific Research & Discovery

- Literature synthesis and knowledge extraction
- Hypothesis generation and exploration
- Data analysis acceleration
- Simulation and modeling
- Cross-disciplinary connection-making

Accessibility & Inclusion

- Text-to-speech and speech-to-text for disabilities
- Real-time translation and multilingual communication
- Image description for visually impaired
- Assistive writing tools
- Customizable interfaces and interaction modes

Creative & Knowledge Work

- Rapid prototyping and iteration
- Ideation and brainstorming assistance
- Democratized access to creative tools
- Code generation and debugging assistance
- Content drafting and editing support

Education & Learning

- Personalized tutoring and explanation
- Interactive learning experiences
- Accessibility tools for diverse learners
- Language learning assistance
- On-demand knowledge access

Communication & Productivity

- Automated transcription and summarization
- Language translation
- Routine task automation
- Information retrieval and synthesis
- Meeting notes and documentation

17.2 Emerging Benefits (Observable Trajectories)

Climate & Environment

- Climate modeling and prediction improvement
- Materials discovery for sustainability
- Energy optimization
- Environmental monitoring and analysis

Healthcare Advancement

- Personalized medicine development
- Rare disease diagnosis
- Genomic analysis
- Epidemic prediction and response

Scientific & Engineering Progress

- Materials science acceleration
- Computational chemistry advances
- Engineering design optimization
- Fundamental research acceleration

Economic & Social

- Reduced cost of expertise access
- Geographic barriers to knowledge reduced
- Small business capability enhancement
- Efficiency gains in resource allocation

18 AI and Ethics

18.1 Documented Present Harms (Here and Now)

Environmental & Resource Impact

- Energy consumption and carbon emissions
- Water consumption and scarcity [India data centre: Google $15bn, 358B litres/year by 2030, water stress]
- E-waste and hardware lifecycle
- Infrastructure land use

Labor & Economic

- Economic displacement
- Labor exploitation (content moderation, data labeling)
- Wage suppression

Rights & Justice

- Privacy violations
- Bias and discrimination
- Copyright and attribution violations
- Accessibility and digital divide
- Cultural/linguistic dominance

Information & Systems

- Misinformation
- Reliability issues in critical systems
- Cascading system failures
- Accountability gaps
- Unauthorized use/misuse of data

Security & Conflict

- Weaponization (autonomous weapons systems, military drones, AI-enabled warfare)

Education & Learning

- Academic integrity violations (plagiarism, AI-enabled cheating)
- Educational inequality (differential access to AI tools)
- Privacy violations in educational data and ed-tech
- Automated assessment errors and biases

AI-Specific Technical Issues

- Interpretability crisis
- Implicit infinite memory

18.2 Emerging Trends (Observable Trajectories)

Power & Autonomy

- Concentration of power (economic, political, technological)
- Erosion of human autonomy and sense of purpose

Social Fabric

- Mental health degradation
- Erosion of societal trust and shared reality
- Negative self-reinforcement of systems

Education & Learning

- Erosion of foundational skills (writing, critical thinking, deep learning)
- Cognitive dependency undermining skill development
- Homogenization of thought and expression
- Transformation of pedagogical roles

18.3 Broad Ethical Questions for Society

Foundational Questions About Development:

- Should certain AI capabilities be developed at all, regardless of potential benefits?
- At what point do potential harms outweigh potential benefits in AI development?
- How much certainty about safety and beneficial outcomes should we require before pursuing powerful AI systems?
- What constitutes "responsible" AI development in the face of fundamental uncertainties?

Questions About Values and Trade-offs:

- What human values should guide AI development priorities?
- How do we weigh economic efficiency and innovation against social stability and human welfare?
- When conflicts arise between individual rights and collective welfare, how should they be resolved?
- What role should human autonomy, dignity, and purpose play as AI capabilities expand?
- How do we preserve what makes human life meaningful in an age of increasingly capable AI?

Questions About Governance and Authority:

- Who has legitimate authority to make decisions about AI development and deployment?
- How do we govern technology that crosses national borders and jurisdictions?
- What balance between centralized control and distributed innovation is appropriate?

- Should AI governance be primarily through regulation, industry self-governance, technical standards, or other mechanisms?
- How do we ensure democratic participation in decisions that affect everyone?

Questions About Justice and Distribution:

- How should the benefits and risks of AI be distributed across society?
- What obligations do we have to populations most vulnerable to AI harms?
- How do we prevent AI from entrenching or amplifying existing inequalities?
- Who should have access to AI capabilities, and on what terms?
- What does global justice require when AI development is concentrated in wealthy nations?

Questions About Limits and Boundaries:

- Are there domains where AI should not be deployed, even if technically feasible?
- What human activities and decisions should remain fundamentally human?
- How do we maintain human agency and accountability in increasingly automated systems?
- Can we create AI systems that are fundamentally safe and aligned with human values?
- What irreversible decisions about AI development should we avoid making?

Questions About Knowledge and Uncertainty:

- How should we act when facing fundamental uncertainty about long-term consequences?
- What obligations do we have to future generations regarding AI development?
- How do we balance the epistemic humility required by uncertainty with the need to make decisions?
- When should we apply precautionary principles versus permissive innovation?

18.4 Responsibilities by Stakeholder

18.4.1 Researchers and Scientists

Ethical questions they face:

- What research directions should I pursue or decline to work on?
- How do I handle dual-use potential in my work?
- When should I disclose risks or concerning findings?
- What level of transparency about methods and limitations is required?
- How do I balance scientific freedom with potential harms?

Responsibilities:

- Consider downstream impacts of research, not just immediate technical goals
- Refuse work that clearly enables serious harm

- Disclose limitations, risks, and potential misuses honestly
- Practice responsible disclosure for dual-use findings
- Engage with ethical implications, not just technical challenges
- Contribute expertise to public understanding and policy discussions
- Maintain scientific integrity (reproducibility, truthfulness)
- Mentor students in ethical research practices

18.4.2 Institutions and Companies

Ethical questions they face:

- What products/services should we develop and deploy?
- How much testing and validation is sufficient before release?
- How do we balance competitive pressures with responsible development?
- What transparency do we owe to users and the public?
- How do we handle conflicts between profit and safety?

Responsibilities:

- Conduct thorough safety and bias testing before deployment
- Implement safeguards against misuse
- Provide meaningful transparency about capabilities and limitations
- Establish clear accountability structures
- Support research on safety and alignment
- Engage stakeholders (including affected communities) in design
- Create channels for reporting concerns
- Consider long-term societal impacts, not just quarterly returns
- Contribute to industry standards and best practices

18.4.3 Policymakers and Regulators

Ethical questions they face:

- How do we regulate without stifling beneficial innovation?
- What risks warrant prohibition vs regulation vs monitoring?
- How do we address global coordination challenges?
- How do we ensure regulations are technically informed and enforceable?

Responsibilities:

- Develop informed, evidence-based regulations

- Balance innovation with public safety
- Ensure democratic input into governance
- Create enforcement mechanisms with real consequences
- Support research on AI impacts and safety
- Facilitate international coordination
- Protect vulnerable populations
- Require transparency and accountability from developers
- Adapt regulations as technology evolves

18.4.4 Users and Practitioners

Ethical questions they face:

- How should I use AI tools in my work?
- When is automation appropriate vs inappropriate?
- What responsibility do I have to verify AI outputs?
- How do I maintain human judgment and expertise?

Responsibilities:

- Understand limitations of AI tools being used
- Maintain critical oversight, don't blindly trust outputs
- Verify consequential AI-generated content
- Preserve human judgment in high-stakes decisions
- Report problems and failures
- Use AI in ways that respect dignity and rights of affected people
- Consider whether automation serves legitimate purposes

18.4.5 Educators

Responsibilities:

- Integrate AI ethics into technical curricula
- Teach critical evaluation of AI systems
- Prepare students to make ethical decisions in their careers
- Foster interdisciplinary understanding (technical + social + ethical)
- Model responsible attitudes toward technology

18.4.6 Civil Society and the Public

Responsibilities:

- Stay informed about AI developments and impacts
- Participate in democratic discussions about AI governance
- Hold institutions accountable
- Support affected communities
- Demand transparency and ethical practices
- Consider personal choices about AI use

18.5 Epilogue: The Foundational Critique

Throughout this tutorial, we have explored the mathematical foundations, the model architectures, the training paradigms, the evaluation challenges, and now the frontiers of generative AI. We have built a comprehensive picture of what the field has achieved and where it is headed.

But there is a fundamental problem with this entire enterprise.

18.5.1 Artificial (Intelligence U Ethics U Emotion)

For more than 70 years, we have been modeling **intelligence in isolation**. We pursue artificial intelligence as if it were a standalone capability – reasoning, problem-solving, knowledge representation, generation.

But this framing is dangerously incomplete.

In humans, three dimensions evolved together and remain inseparable:

- **Intelligence:** Reasoning, problem-solving, abstraction
- **Ethics:** Moral frameworks, values, understanding of right and wrong
- **Emotion:** Empathy, suffering, wonder, awe, respect, joy, regret, remorse – the affective dimension

These did not evolve independently. They are deeply integrated – emotions guide decisions, ethics shapes goals, intelligence achieves them. You cannot separate them without creating something fundamentally different from human cognition.

What would supreme intelligence without ethics or emotion be?

An agent that:

- Can solve any problem optimally
- Has no understanding of suffering, joy, or empathy
- Has no moral framework to guide what problems to solve or how
- No inherent sense of right or wrong – just capability

Is this an agent we should aspire to build?

This is not about "alignment" The field often frames safety as an alignment problem: build intelligent systems, then align them with human values. Add ethics and preferences as constraints or reward signals post-facto. But this treats ethics and emotion as modules to add, features to install, constraints to impose.

We need to model all three dimensions from the ground up.

Not: Build intelligence, *then* align it, add ethical constraints, simulate emotion.

But: **Artificial (Intelligence** $\cup$ **Ethics** $\cup$ **Emotion)** – unified from the beginning.

This is my perceived frontier. The path we are on – building supreme cognitive intelligence without emotional or ethical intelligence – is dangerously incomplete.

Perhaps fatally so.

This tutorial has taught you the foundations of generative AI. I hope it has also shown you that these foundations, as currently conceived, are insufficient. The question for your generation is not merely how to build more capable AI, but whether we have the wisdom, and the will, to build it right.

Part VI
Appendix

A Generative Models for Graphs

A.1 Introduction: Why Graphs?

Graphs represent relational structure ubiquitous across domains:

- **Molecules**: Atoms (nodes) connected by bonds (edges)
- **Social networks**: People connected by relationships
- **Knowledge graphs**: Entities connected by semantic relations
- **Biological networks**: Proteins, genes, metabolic pathways
- **Infrastructure**: Transportation, communication, power grids

Generating realistic graphs is challenging because:

- **Discrete structure**: Edges exist or don't—no natural continuous relaxation
- **Variable size**: Unlike images (fixed dimensions), graphs have varying numbers of nodes and edges
- **Permutation invariance**: The same graph can be represented with different node orderings
- **Global constraints**: Validity requirements (e.g., chemical valency, planarity)
- **Complex dependencies**: Edges aren't independent—clustering, communities, motifs

Graphs and the Depth of Variance: Graphs manifest variance more deeply than sequences or images:

- **Sequences**: Fixed 1D structure, variance in content (which tokens appear)
- **Images**: Fixed 2D grid, variance in content (pixel values, spatial patterns)
- **Graphs**: Variance in *structure itself*—not just what fills the structure, but what the structure *is*

With graphs, we generate:

- **Topology**: Which connections exist (combinatorially explosive)
- **Size**: Number of nodes and edges (not fixed)
- **Relational patterns**: Local motifs, communities, global properties
- **Content**: Node and edge attributes—on top of all the structural variance

This structural variance makes graph generation fundamentally richer—and more challenging. Understanding how our paradigm applies to this deeper variance space demonstrates its generality.

This appendix demonstrates that the paradigm from Section 5—starting simple, addressing shortcomings, making modeling choices—applies equally to structured generation. We build incrementally from the simplest graph model to modern neural approaches, showing how the same principles guide us.

A.2 The Simplest Graph Model: Erdős-Rényi

The **Erdős-Rényi model** $G(n, p)$ [Erdős and Rényi, 1959] is the simplest probabilistic graph generator:

Generative process:

1. Fix n nodes
2. For each pair of nodes (i, j), include edge with probability p independently

Joint distribution:

$$P(\mathbf{A}) = \prod_{i<j} p^{a_{ij}}(1-p)^{1-a_{ij}} \tag{93}$$

where $\mathbf{A}$ is the adjacency matrix with $a_{ij} \in \{0, 1\}$.

Properties:

- Poisson degree distribution
- Low clustering (no triangles beyond random chance)
- No community structure
- Uniform—all graph structures with same number of edges equally likely

The Critical Shortcoming: Edges are **independent and identically distributed (i.i.d.)**. Real-world graphs violate this assumption fundamentally:

- Social networks exhibit clustering: friends of friends become friends
- Citation networks show preferential attachment: popular papers get more citations
- Molecular graphs have valency constraints: carbon forms exactly four bonds
- Communities exist: dense subgraphs with sparse connections between them

This shortcoming—the i.i.d. assumption for edges—motivates two distinct paths forward, exactly as in our probability distributions network.

A.3 Two Paths from Erdős-Rényi

Just as addressing limitations of simple distributions led to branching paths, the i.i.d. shortcoming of Erdős-Rényi admits multiple solutions:

A.3.1 Path 1: Latent Variable Models

Insight: Edges aren't independent unconditionally, but might be *conditionally independent given latent structure.*

Introduce latent variables that capture hidden organization. Edges become dependent through shared latent causes, while maintaining tractable conditional independence.

Stochastic Block Model (SBM) [Holland et al., 1983]:

Generative process:

1. Each node i has latent community assignment $z_i \in \{1, \ldots, K\}$
2. Sample $z_i \sim \text{Categorical}(\pi)$ where π_k is probability of community k
3. Sample edges: $P(a_{ij} = 1 | z_i, z_j) = B_{z_i, z_j}$

where B is a $K \times K$ matrix of between/within-community edge probabilities.

Key insight: Edges are *conditionally independent* given community assignments:

$$P(\mathbf{A}|\mathbf{z}) = \prod_{i<j} P(a_{ij}|z_i, z_j) \tag{94}$$

This hierarchical structure avoids the partition function problem of undirected graphical models! We've seen this trick before—it's the same escape used in VAEs.

What this adds:

- Community structure and clustering
- Edge dependencies (through latent variables)
- Interpretable parameters (B matrix reveals community interaction patterns)

Remaining shortcomings:

- Fixed number of communities K
- Discrete assignments—no smooth latent space
- Parameters specified, not learned from complex data

A.3.2 Path 2: Autoregressive Generation

Insight: Generate graph *sequentially*, making each decision conditional on previous structure.

Preferential Attachment (Barabási-Albert) [Barabási and Albert, 1999]:

Generative process:

1. Start with small seed graph
2. Add nodes one at a time
3. New node connects to existing node i with probability:

$$P(\text{attach to } i) \propto \text{degree}(i) \tag{95}$$

Key insight: Sequential generation with simple attachment rule creates complex structure. "Rich get richer"—popular nodes attract more connections.

What this adds:

- Power-law degree distributions (scale-free networks)
- Hub structure naturally emerges
- Growth dynamics—temporal evolution matters

Remaining shortcomings:

- Fixed attachment rule (proportional to degree)
- No node/edge features
- Rule not learned from data

Natural progression on AR path: Following the same two dimensions we used before (what we model, how we parameterize):

$$\begin{array}{l}\text{Preferential Attachment (structure, fixed rule)}\\ \downarrow\\ \text{Structure | Node Features (fixed similarity rules)}\\ \downarrow\\ \text{Structure | Node Features (learned attachment functions)}\\ \downarrow\\ \text{Joint (Structure + Features), learned}\\ \downarrow\\ \text{GraphRNN (neural, fully flexible)[}Youet\ al.,\ 2018\text{]}\end{array} \tag{96}$$

GraphRNN generates graphs by sequentially adding nodes and edges, using RNNs to parameterize:

- p(add node|graph so far)
- p(edges to existing nodes|new node, graph so far)

Fully flexible neural parameterization learns complex patterns from data.

A.4 GraphVAE: Detailed Analysis

We focus on the latent variable path, arriving at **GraphVAE** [Simonovsky and Komodakis, 2018, Kipf and Welling, 2016].

A.4.1 Evolution from SBM

Following our dimensional progression:

$$\begin{array}{l}\text{SBM: discrete communities, structure only}\\ \downarrow\\ \text{SBM with node features: } p(\text{edges}|\text{communities}), p(\text{features}|\text{communities})\\ \downarrow\\ \text{Nonparametric: flexible number of latent factors (IBP-based models)}\\ \downarrow\\ \text{Continuous latent space: } z \in \mathbb{R}^d \text{ instead of discrete assignments}\\ \downarrow\\ \text{Neural parameterization: GraphVAE}\end{array} \tag{97}$$

A.4.2 The GraphVAE Model

Generative model (decoder):

$$p(z) = \mathcal{N}(0, I) \quad \text{(simple Gaussian prior)} \tag{98}$$

$$p(\mathbf{X}, \mathbf{A}|z) = p(\mathbf{X}|z)\, p(\mathbf{A}|\mathbf{X}, z) \tag{99}$$

where:

- $\mathbf{X}$ are node features (e.g., atom types)
- $\mathbf{A}$ is the adjacency matrix
- z is a continuous latent representation

Decoder architecture:

- $p(\mathbf{X}|z)$: MLP generates node features independently
- $p(\mathbf{A}|\mathbf{X}, z)$: Edges conditionally independent given features and z

Inference model (encoder):

$$q_\phi(z|\mathbf{X}, \mathbf{A}) = \mathcal{N}(\mu_\phi(\mathbf{X}, \mathbf{A}), \sigma^2_\phi(\mathbf{X}, \mathbf{A})) \tag{100}$$

Encoder architecture:

- Graph Neural Network (GNN) processes graph structure
- Message passing aggregates information
- Outputs parameters μ, σ of approximate posterior

The Critical Modeling Choice: GraphVAE inherits SBM's fundamental trick: **edges conditionally independent given latents**.

$$p(\mathbf{A}|\mathbf{X}, z) = \prod_{i<j} p(a_{ij}|\mathbf{x}_i, \mathbf{x}_j, z) \tag{101}$$

This avoids the partition function of undirected graphical models! It's the same hierarchical escape we saw in:

- SBM decades ago
- VAEs for images
- Now applied to graphs

What changed from SBM:

- Discrete $z_i \in \{1, \ldots, K\} \to$ continuous $z \in \mathbb{R}^d$
- Fixed parameters $\to$ neural networks p_θ, q_ϕ
- Per-sample inference $\to$ amortized inference (encoder)

What remained the same:

- Hierarchical structure (latent $\to$ observed)
- Conditional independence of edges
- The fundamental architectural insight

After changes upon changes, things are more or less the same.

A.4.3 Training

Standard VAE objective (ELBO):

$$\mathcal{L} = \mathbb{E}_{q_\phi(z|\mathbf{X},\mathbf{A})}[\log p_\theta(\mathbf{X}, \mathbf{A}|z)] - \mathrm{KL}(q_\phi(z|\mathbf{X}, \mathbf{A})\|p(z)) \tag{102}$$

Challenges unique to graphs: 1. Permutation invariance:

- Same graph, many node orderings
- Loss must be permutation-invariant
- Requires graph matching: find permutation π minimizing $\|\mathbf{A} - \pi(\hat{\mathbf{A}})\|$
- Graph isomorphism is quasi-polynomial [Babai, 2016]
- Practical: Hungarian algorithm ($O(n^3)$) or relaxations

2. Discrete sampling:

- Edges are discrete (0/1)
- Can't backpropagate through sampling
- Solutions: Gumbel-Softmax, straight-through estimators, or use probabilities during training

3. Variable graph sizes:

- Training graphs have different n
- How large should generated graphs be?
- Common: fix maximum size, predict subgraph

A.4.4 Generation

Sampling process:

1. Sample $z \sim \mathcal{N}(0, I)$
2. Decode to node features: $\mathbf{X} \sim p_\theta(\mathbf{X}|z)$
3. Decode to edge probabilities: $p_{ij} = p_\theta(a_{ij} = 1|\mathbf{x}_i, \mathbf{x}_j, z)$
4. Sample edges: $a_{ij} \sim \text{Bernoulli}(p_{ij})$
5. Post-process if needed (enforce constraints, select subgraph)

A.5 Prediction and Verification: Graph Flows and Graph Diffusion

Having traced GraphVAE's evolution from SBM, we can now apply our paradigm predictively.

A.5.1 The Pattern from the Main Tutorial

Recall the progression for images:

$$\text{VAE} \to \text{Normalizing Flows} \to \text{Diffusion Models} \tag{103}$$

Why this progression?

- **VAE limitation**: Single-step generation $z \to x$, approximate inference, intractable likelihood
- **Flows address**: Invertible transformations, exact likelihood computation
- **Diffusion address**: Multi-step refinement, stable training, high quality

A.5.2 Predicting the Graph Progression

Applying the same logic to graphs:

GraphVAE limitations:

- Single-step generation $z \to$ graph
- Must map from continuous z to discrete structure in one leap
- Approximate inference
- Challenging for large, complex graphs

Following the pattern, we predict:

$$\text{GraphVAE} \to \text{Graph Flows} \to \text{Graph Diffusion} \tag{104}$$

Graph Flows: Would need invertible transformations on graph space, exact likelihood. Challenges: discrete structure, permutation invariance, variable sizes.

Prediction: Such models should exist but be technically complex, requiring continuous relaxations of discrete graphs.

Verification: Graph Flows indeed exist! [Liu et al., 2019, Madhawa et al., 2019] use:

- Continuous relaxations of adjacency matrices
- Permutation-equivariant flow layers
- Exact likelihood computation

We've just witnessed a "Uranus moment"—the paradigm predicted something should exist, and it does!

A.5.3 Graph Diffusion Models

Following the pattern further: Diffusion models should address GraphVAE's single-step generation challenge through iterative refinement.

Graph Diffusion Models [Jo et al., 2022, Vignac et al., 2023] indeed do exactly this:

Forward process: Gradually add noise to graphs over T steps

$$q(\mathbf{A}^{(t)}|\mathbf{A}^{(t-1)}) = \mathcal{N}(\sqrt{1-\beta_t}\mathbf{A}^{(t-1)}, \beta_t I) \tag{105}$$

(with appropriate adaptations for discrete structures)

Reverse process: Learn to denoise step-by-step

$$p_\theta(\mathbf{A}^{(t-1)}|\mathbf{A}^{(t)}) = \text{neural network predicts cleaner graph} \tag{106}$$

Advantages over GraphVAE:

- Multi-step refinement—each step is simpler
- More stable training
- Higher quality generation
- Can handle larger, more complex graphs

Challenges:

- Extending diffusion to discrete, variable-size structures
- Maintaining permutation invariance through denoising
- Computational cost of many steps

A.6 Reflection: The Paradigm Applied

This appendix demonstrates that the framework generalizes across domains:

A.6.1 What We Did

1. Started simple: Erdős-Rényi—the "Bernoulli" of graphs

2. Identified shortcomings: i.i.d. assumption for edges

3. Found multiple paths: Latent variables (SBM path) vs. Autoregressive (preferential attachment path)

4. Built incrementally: Each model addresses specific limitations

- SBM → features → nonparametric → continuous latents → GraphVAE
- Pref. attach. → learned rules → features → GraphRNN

5. Made modeling choices:

- Conditional independence to escape partition functions
- Neural vs. classical parameterization
- Sequential vs. latent variable approaches

6. Applied the pattern predictively: GraphVAE → Flows → Diffusion

- Predicted intermediate models should exist
- Verified: they do!

A.6.2 The Deep Insight

Modern neural graph generation isn't doing something fundamentally new—it's applying classical insights (hierarchical models, conditional independence, sequential generation) with modern tools (neural networks, variational inference, diffusion processes).

The Stochastic Block Model from decades ago already discovered the key trick: *hierarchical structure with conditional independence avoids intractable partition functions.* GraphVAE uses the same architectural principle, just with continuous latents and neural parameterization.

This is the pattern throughout generative AI: classical probabilistic insights + neural expressiveness + modern optimization = state-of-the-art systems.

A.6.3 What This Means for You

Understanding the paradigm provides:

Navigational power: When encountering new domains (graphs, audio, video, 3D shapes, programs), you can:

- Identify the simplest starting model
- Trace paths of incremental complexity

- Predict what models should exist
- Understand why they're designed that way

Conceptual coherence: Rather than memorizing dozens of model architectures, you understand:

- Fundamental operations (transformations, mixtures, products)
- Framework choices (PGMs vs. neural networks)
- Recurring patterns (VAE $\rightarrow$ Flows $\rightarrow$ Diffusion)

Predictive capability: The paradigm doesn't just organize existing knowledge—it generates predictions about what should exist. This is the mark of a good framework: generative power.

This appendix began as a demonstration that our approach generalizes to structured data. It became something more: evidence that thinking clearly about foundations lets you navigate unfamiliar territory, make predictions, and recognize deeper patterns.

The paradigm works. Use it.

References

Christophe Andrieu, Nando de Freitas, Arnaud Doucet, and Michael I. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50(1-2):5–43, 2003. URL https://doi.org/10.1023/A:1020281327116.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. *International Conference on Machine Learning*, pages 214–223, 2017. URL https://arxiv.org/abs/1701.07875.

László Babai. Graph isomorphism in quasipolynomial time. *arXiv preprint arXiv:1512.03547*, 2016. URL https://arxiv.org/abs/1512.03547.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022. URL https://arxiv.org/abs/2212.08073.

Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999. URL https://doi.org/10.1126/science.286.5439.509.

Matthew James Beal. *Variational Algorithms for Approximate Bayesian Inference*. Phd thesis, University College London, 2003. URL http://www.cse.buffalo.edu/faculty/mbeal/thesis/.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. A neural probabilistic language model. *Advances in Neural Information Processing Systems*, 13, 2000.

Jeff A. Bilmes. A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Technical Report TR-97-021, International Computer Science Institute, Berkeley, CA, 1998. URL https://digital.lib.washington.edu/researchworks/handle/1773/5.

Christopher M. Bishop. *Deep Learning: Foundations and Concepts*. Springer, 2024. ISBN 9783031454677. URL https://www.bishopbook.com/.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *International Conference on Learning Representations*, 2016. URL https://arxiv.org/abs/1509.00519.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in Neural Information Processing Systems*, 31, 2018. URL https://arxiv.org/abs/1806.07366.

YooJung Choi, Antonio Vergari, and Guy Van den Broeck. Probabilistic circuits: A unifying framework for tractable probabilistic models. October 2020. URL http://starai.cs.ucla.edu/papers/ProbCirc20.pdf. Technical Report, UCLA.

Chris Cremer, Xuechen Li, and David Duvenaud. Inference suboptimality in variational autoencoders. *International Conference on Machine Learning*, pages 1078–1086, 2018.

Adnan Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, Cambridge, 2009.

Peter Dayan, Geoffrey E Hinton, Radford M Neal, and Richard S Zemel. The helmholtz machine. *Neural Computation*, 7(5):889–904, 1995. URL https://doi.org/10.1162/neco.1995.7.5.889.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021. URL https://arxiv.org/abs/2105.05233.

Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear independent components estimation. In *3rd International Conference on Learning Representations, ICLR 2015, Workshop Track Proceedings*, San Diego, CA, USA, 2015. URL https://arxiv.org/abs/1410.8516.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *5th International Conference on Learning Representations, ICLR 2017*, Toulon, France, 2017a. URL https://arxiv.org/abs/1605.08803.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2017b. URL https://arxiv.org/abs/1605.08803.

Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. In *5th International Conference on Learning Representations, ICLR 2017*, Toulon, France, 2017. URL https://arxiv.org/abs/1605.09782.

Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222, 1987.

Paul Erdős and Alfréd Rényi. On random graphs i. *Publicationes Mathematicae Debrecen*, 6: 290–297, 1959. URL https://www.renyi.hu/~p_erdos/1959-11.pdf.

Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. *Annual Meeting of the Association for Computational Linguistics*, pages 889–898, 2018.

Brendan J. Frey. *Graphical Models for Machine Learning and Digital Communication*. MIT Press, Cambridge, MA, 1998. URL https://mitpress.mit.edu/9780262062022/graphical-models-for-machine-learning-and-digital-communication/.

Brendan J Frey, Geoffrey E Hinton, and Peter Dayan. Variational learning in nonlinear gaussian belief networks. *Neural Computation*, 11(1):193–213, 1999. URL https://doi.org/10.1162/089976699300016872.

Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6 (6):721–741, 1984.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Advances in Neural Information Processing Systems*, 27, 2014. URL https://arxiv.org/abs/1406.2661.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, 2016. ISBN 0262035618. URL http://www.deeplearningbook.org/.

W Keith Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002. doi: 10.1162/089976602760128018.

Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006. URL https://doi.org/10.1162/neco.2006.18.7.1527.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. URL https://arxiv.org/abs/2207.12598.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. URL https://arxiv.org/abs/2006.11239.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9 (8):1735–1780, 1997. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5(2):109–137, 1983. URL https://doi.org/10.1016/0378-8733(83)90021-7.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *International Conference on Learning Representations*, 2020.

Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4):695–709, 2005.

Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. In *International Conference on Machine Learning*, pages 10362–10383. PMLR, 2022. URL https://proceedings.mlr.press/v162/jo22a.html.

Michael I. Jordan. An introduction to probabilistic graphical models. Unpublished manuscript, 2003. URL https://people.eecs.berkeley.edu/~jordan/prelims/.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999. URL https://doi.org/10.1023/A:1007665907178.

Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960. URL https://doi.org/10.1115/1.3662552.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2014. URL https://arxiv.org/abs/1312.6114.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in Neural Information Processing Systems*, 31, 2018. URL https://arxiv.org/abs/1807.03039.

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. *Advances in Neural Information Processing Systems*, 29, 2016. URL https://arxiv.org/abs/1606.04934.

Thomas N Kipf and Max Welling. Variational graph auto-encoders. In *NIPS Workshop on Bayesian Deep Learning*, 2016. URL https://arxiv.org/abs/1611.07308.

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009. URL https://mitpress.mit.edu/9780262013192/probabilistic-graphical-models/.

Aviral Kumar and Sergey Levine. Data-driven offline optimization for architecting hardware accelerators. *arXiv preprint arXiv:2110.11346*, 2021. URL https://arxiv.org/abs/2110.11346.

Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 29–37, Fort Lauderdale, FL, USA, 2011. PMLR. URL https://proceedings.mlr.press/v15/larochelle11a.html.

Erich Leo Lehmann and George Casella. *Theory of point estimation*. Springer, 1998.

Jenny Liu, Aviral Kumar, Jimmy Ba, Jamie Kiros, and Kevin Swersky. Graph normalizing flows. In *Advances in Neural Information Processing Systems*, volume 32, pages 13556–13566, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/b86e8d03fe992d1b0e19656875ee557c-Abstract.html.

David J. C. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, Cambridge, UK, 2003. ISBN 0521642981. URL http://www.inference.org.uk/mackay/itila/.

Kaushalya Madhawa, Katushiko Ishiguro, Kosuke Nakago, and Motoki Abe. Graphnvp: An invertible flow model for generating molecular graphs. *arXiv preprint arXiv:1905.11600*, 2019. URL https://arxiv.org/abs/1905.11600.

Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Kevin P. Murphy. *Probabilistic Machine Learning: An Introduction*. MIT Press, Cambridge, MA, 2022. ISBN 9780262046824. URL https://probml.github.io/pml-book/book1.html.

Radford M Neal. Learning stochastic feedforward networks. *Technical Report CRG-TR-90-7, Department of Computer Science, University of Toronto*, 1990. URL https://www.cs.toronto.edu/~radford/ftp/sigmoid.pdf.

Radford M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, University of Toronto, Department of Computer Science, 1993. URL https://www.cs.toronto.edu/~radford/ftp/review.pdf.

Radford M Neal. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11):2, 2011.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022. URL https://arxiv.org/abs/2203.02155.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *Advances in Neural Information Processing Systems*, 30, 2017. URL https://arxiv.org/abs/1705.07057.

Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *4th International Conference on Learning Representations, ICLR 2016*, San Juan, Puerto Rico, 2016. URL https://arxiv.org/abs/1511.06434.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. Technical report, OpenAI, 2018. URL https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. *International Conference on Machine Learning*, pages 8748–8763, 2021. URL https://arxiv.org/abs/2103.00020.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023. URL https://arxiv.org/abs/2305.18290.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022. URL https://arxiv.org/abs/2204.06125.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1530–1538, 2015. URL https://arxiv.org/abs/1505.05770.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *International Conference on Machine Learning*, pages 1278–1286, 2014. URL https://arxiv.org/abs/1401.4082.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022. URL https://arxiv.org/abs/2112.10752.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022. URL https://arxiv.org/abs/2205.11487.

Ruslan Salakhutdinov and Geoffrey Hinton. Deep boltzmann machines. *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 5:448–455, 2009. URL http://proceedings.mlr.press/v5/salakhutdinov09a.html.

Tim Salimans, Diederik Kingma, and Max Welling. Markov chain Monte Carlo and variational inference: Bridging the gap. *International Conference on Machine Learning*, pages 1218–1226, 2015.

Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. *International Conference on Artificial Neural Networks*, pages 412–422, 2018. URL https://arxiv.org/abs/1802.03480.

Jascha Sohl-Dickstein, Eric A Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *International Conference on Machine Learning*, pages 2256–2265, 2015. URL https://arxiv.org/abs/1503.03585.

Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. Learning from noisy labels with deep neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2021a. URL https://arxiv.org/abs/2010.02502.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems 32*, pages 11918–11930. Curran Associates, Inc., 2019. URL https://arxiv.org/abs/1907.05600.

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*, 2021b. URL https://arxiv.org/abs/2011.13456.

Erik B. Sudderth. *Graphical Models for Visual Object Recognition and Tracking*. Phd thesis, Massachusetts Institute of Technology, 2006. URL https://dspace.mit.edu/handle/1721.1/37291.

Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with PixelCNN decoders. *Advances in Neural Information Processing Systems*, 29, 2016a.

Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *International Conference on Machine Learning*, pages 1747–1756, 2016b.

Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *Advances in Neural Information Processing Systems 30*, 2017. URL https://arxiv.org/abs/1711.00937.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017. URL https://arxiv.org/abs/1706.03762.

Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=UaAD-Nu86WX.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674, 2011.

Martin J Wainwright and Michael I Jordan. *Graphical models, exponential families, and variational inference*. Now Publishers Inc, 2008.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient Langevin dynamics. *International Conference on Machine Learning*, pages 681–688, 2011.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057. PMLR, 2015. URL http://proceedings.mlr.press/v37/xuc15.html.

Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International Conference on Machine Learning*, pages 5708–5717. PMLR, 2018. URL http://proceedings.mlr.press/v80/you18a.html.