Chapter 1

Introduction

Over the past decade, artificial intelligence (AI) has undergone a transformative evolution, driven by groundbreaking advancements in generative modeling and self-supervised learning (SSL). These two fields have redefined our ability to model complex data distributions and leverage vast amounts of unlabeled data, leading to significant breakthroughs in areas such as media synthesis, multimodal learning, and scientific discovery.

Generative modeling focuses on learning the underlying probability distributions of data to create realistic and diverse samples. Innovations such as Generative Adversarial Networks (GANs), diffusion models, and normalizing flows have set new benchmarks in quality and applicability, enabling realistic image generation, lifelike video creation, and natural-sounding audio synthesis. Beyond creative media, these models play a critical role in scientific applications, such as molecular modeling, protein structure prediction, and AI-assisted drug discovery. By generating novel molecules and exploring chemical spaces, generative models are accelerating research in fields like biotechnology and materials science. Furthermore, they address challenges in inverse problems, including image super-resolution and inpainting, and excel in multimodal tasks like text-to-image generation.

Self-supervised learning, on the other hand, has emerged as a powerful approach to harness the vast reservoirs of unlabeled data by uncovering its inherent structure. SSL enables models to learn meaningful representations that are crucial for downstream tasks, often serving as the backbone of state-of-the-art systems across diverse domains. These representations can be extracted using generative methods, which model the data distribution, or discriminative methods, such as contrastive learning, which focus on distinguishing between similar and dissimilar data samples.

Self-supervised learning has had a profound impact on foundational technologies. In natural language processing, it powers groundbreaking models like BERT and GPT, which underpin conversational AI and advanced language understanding systems. In computer vision, SSL reduces dependence on labeled data while delivering robust, scalable visual representations, driving improvements in tasks like object recognition and image segmentation. These advancements underscore SSL's versatility and its pivotal role in advancing AI across domains and modalities. Additionally, SSL has been transformative in multimodal learning, enabling seamless alignment and integration across modalities such as text, images, and audio. This capability has unlocked advanced tasks like generating descriptive image captions, synchronizing audio with video, retrieving relevant content across formats, and enhancing representation learning for multimodal datasets.

In this thesis, we focus on the theoretical and computational aspects of generative modeling and generative self-supervised learning. To provide the necessary background, the remainder of this introduction is divided into two sections: an overview of generative modeling (1.1) and an overview of self-supervised learning (1.2).

1.1 Generative Modeling

1.1.1 The Problem of Generative Modeling

Generative modeling sits at the forefront of machine learning, enabling us to capture the underlying structure of complex data and generate new, realistic samples. Whether it's creating lifelike images, composing music, or designing novel molecules, generative models aim to understand and replicate the distribution of data they're trained on.

Formally, let's consider a dataset $\mathscr{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \subset \mathbb{R}^d$, comprising N independent samples drawn from an unknown data-generating distribution P, which we refer to as the *target distribution*. The central goal of generative modeling is to create a model that approximates P as closely as possible based solely on these samples.

To learn the distribution P, we seek a computationally tractable approximation $\hat{P}$ based on the dataset $\mathscr{D}$ and our assumptions about P. We measure how close $\hat{P}$ is to P using a divergence $D(\hat{P}, P)$, such as the Kullback-Leibler divergence or the Wasserstein distance. Our goal is to minimize this divergence:

$$\hat{P} = \arg\min_{\hat{P}} D(\hat{P}, P).$$

Typically, we parameterize $\hat{P}$ with a finite-dimensional vector $\theta \in \mathbb{R}^p$, forming a family of models $\hat{P}_\theta$ and transforming the problem into optimizing over parameters:

$$\theta^* = \arg\min_{\theta} D(\hat{P}_\theta, P).$$

For practicality, $\hat{P}_\theta$ must be computationally tractable; we should be able to efficiently sample from it or evaluate its probability density function. Different generative modeling approaches balance tractability and fidelity to P, leading to various methods suitable for different applications.

By tackling this problem, we aim not only to reproduce the data we observe but also to uncover the underlying mechanisms that generate it. This deep understanding empowers us to create models that can generate new, unseen data that is indistinguishable from real-world samples, thereby expanding the horizons of what machines can learn and create.

1.1.2 Why Deep Learning for Generative Modeling?

Non-parametric methods such as Kernel Density Estimation (KDE) have been widely used for modeling probability distributions in low-dimensional settings due to their flexibility in estimating the underlying density without assuming a specific parametric form [56]. However, scaling KDE to high-dimensional data is problematic. The sample complexity of KDE scales exponentially with the number of feature dimensions d, making it impractical for high-dimensional tasks [65]. Specifically, to achieve a target estimation error ε, the number of samples n must satisfy:

$$n \geq \mathcal{O}\left(\varepsilon^{-\frac{4+d}{4}}\right).$$

[55] This exponential dependence on d implies that even for moderate dimensions, the required sample size becomes infeasibly large, rendering KDE and similar methods ineffective for generative modeling in high dimensions.

In many real-world applications, however, data does not uniformly occupy the high-dimensional ambient space but is concentrated near a lower-dimensional manifold—a concept known as the *manifold hypothesis* [16]. For example, natural images, although high-dimensional when represented by pixel values, often lie on manifolds of much lower intrinsic dimensionality due to dependencies and correlations among pixels [52]. This implies that the effective complexity of the data is determined by the manifold's dimension K, where $K \ll d$.

While the manifold hypothesis suggests a reduction in effective dimensionality, leveraging this property requires models that can explicitly capture the underlying manifold structure. Non-parametric methods like KDE do not inherently account for the manifold and treat each dimension independently, failing to exploit the dependencies among features [62]. As a result, they still suffer from high sample complexity despite the data's lower intrinsic dimensionality.

Parametric models with appropriate inductive biases can overcome this limitation by incorporating assumptions about the data's structure into the learning process. Deep learning architectures are particularly well-suited for this task due to their ability to integrate domain-specific inductive biases through their design [33]. For instance, Convolutional Neural Networks (CNNs) employ local connectivity and weight sharing to capture spatial hierarchies and local patterns in image data [34]. Similarly, Recurrent Neural Networks (RNNs) are designed to model temporal dependencies in sequential data [27].

By aligning model architecture with the intrinsic properties of the data manifold, deep learning models effectively reduce the hypothesis space to functions consistent with the data's structure, enhancing learning efficiency and reducing the required number of samples [46]. This structural approach explains why deep learning–based methods have become indispensable for generative modeling in high dimensions [18, 31].

1.1.3 Deep Learning for Generative Modeling

In the past decade, deep learning has introduced flexible architectures that can effectively learn complex data distributions. This chapter reviews five main types of generative models: Generative Adversarial Networks (GANs), Normalizing Flows, Auto-Regressive Models, Energy-Based Models, and Diffusion Models, each with unique strengths and limitations.

Generative Adversarial Networks (GANs)

Introduced by Goodfellow et al. [19], Generative Adversarial Networks (GANs) consist of two neural networks: a *generator* G and a *discriminator* D, trained simultaneously in a minimax game. The generator maps samples $\mathbf{z}$ from a simple prior distribution $P_{\mathbf{z}}$ (e.g., a standard normal distribution) to the data space $\mathbb{R}^d$, producing synthetic data $G(\mathbf{z})$. The discriminator outputs a probability $D(\mathbf{x}) \in [0,1]$ indicating whether a sample $\mathbf{x}$ is real (from the training data) or generated (from G).

The training objective is formulated as a two-player minimax game with the value function $V(D,G)$:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim P_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim P_{\mathbf{z}}}[\log(1 - D(G(\mathbf{z})))]$$

where P_{data} is the data distribution represented by the training set $\mathscr{D}$.

Training Process GANs are trained through an adversarial process where the generator and discriminator are updated iteratively to improve against each other. The discriminator

D learns to better distinguish real data from generated data, while the generator G aims to produce data that can fool D. This training continues until the generator produces samples indistinguishable from real data ($P_G = P_{\text{data}}$).

There are various formulations of the value function designed to enhance training stability and performance. For instance, the Wasserstein GAN (WGAN) [2] modifies the original objective using the Earth Mover's distance to address issues like mode collapse and improve convergence.

Strengths GANs have several notable strengths. One of their primary advantages is their ability to produce **high-quality samples** that are visually realistic and sharp, especially in image generation tasks. Additionally, GANs offer **efficient sampling**; once the generator is trained, it can rapidly produce new samples through a single forward pass, without the need for iterative sampling procedures common in other generative models.

Weaknesses Despite their strengths, GANs also present significant weaknesses. A major challenge is the **training instability** inherent in their adversarial setup, which can lead to issues such as mode collapse—where the generator produces limited varieties of samples—vanishing gradients, and oscillatory behavior during training. Moreover, GANs **do not provide an explicit density** function or likelihood estimate for the generated data, limiting their applicability in tasks that require probabilistic interpretations or quantitative assessments of sample likelihood.

Normalizing Flows

Normalizing flows, introduced by Rezende and Mohamed [50], are a class of generative models that model complex probability distributions by transforming a simple base distribution through a sequence of invertible and differentiable mappings. The core idea is to start with a tractable distribution, such as a standard Gaussian, and apply a series of transformations to model the target data distribution accurately.

Given a latent variable $Z \sim p_Z$ in latent space $\mathscr{Z}$ and an invertible function $G^\theta : \mathscr{Z} \to \mathscr{X}$ parameterized by θ, the model defines the random variable $X = G^\theta(Z)$ in data space $\mathscr{X}$ to approximate the data distribution p_X. The transformation G^θ maps samples from the base distribution to samples resembling the observed data.

Change of Variables To compute the probability density function (pdf) of X, the change of variables formula is employed. For an invertible and differentiable transformation G^θ, with inverse $F^\theta = (G^\theta)^{-1}$, the pdf of X is given by:

$$p_X^\theta(\mathbf{x}) = p_Z\left(F^\theta(\mathbf{x})\right)\left|\det\left(\frac{\partial F^\theta(\mathbf{x})}{\partial \mathbf{x}}\right)\right|,$$

where $\frac{\partial F^\theta(\mathbf{x})}{\partial \mathbf{x}}$ is the Jacobian matrix of F^θ at point $\mathbf{x}$, and det denotes the determinant. Taking the logarithm yields:

$$\log p_X^\theta(\mathbf{x}) = \log p_Z\left(F^\theta(\mathbf{x})\right) + \log\left|\det\left(\frac{\partial F^\theta(\mathbf{x})}{\partial \mathbf{x}}\right)\right|.$$

This formulation enables exact computation of the log-likelihood, which is crucial for effective training.

Composing Transformations In practice, the transformation G^θ is constructed by composing a sequence of K invertible and differentiable transformations:

$$G^\theta = G_K \circ G_{K-1} \circ \cdots \circ G_1,$$

where each G_k is an invertible function, and the composition is defined as $G^\theta = G_K(G_{K-1}(\ldots G_1(z)))$. The inverses of these transformations are denoted as $F_k = G_k^{-1}$.

For a data point $\mathbf{x}$, intermediate variables are defined by recursively applying the inverses:

$$\mathbf{h}_0 = \mathbf{x}, \quad \mathbf{h}_k = F_k(\mathbf{h}_{k-1}), \quad \text{for } k = 1, \ldots, K.$$

The log-density of $\mathbf{x}$ can then be expressed as:

$$\log p_X^\theta(\mathbf{x}) = \log p_Z(\mathbf{h}_K) + \sum_{k=1}^{K} \log\left|\det\left(\frac{\partial F_k(\mathbf{h}_{k-1})}{\partial \mathbf{h}_{k-1}}\right)\right|.$$

Efficient Computation A crucial aspect of designing normalizing flows is ensuring that the composing functions G_k are, typically, analytically invertible to allow for fast sampling. Additionally, the calculation of the determinant of the Jacobian must be tractable to enable efficient training via the change of variables formula. This is often achieved by designing the transformations so that the resulting Jacobian matrices have a lower triangular form or another structure that simplifies the computation of the determinant. By constructing G_k in this manner, normalizing flows facilitate efficient evaluation of both the forward and inverse transformations, as well as the log-determinant of the Jacobian, thereby enabling effective training and fast sampling from the model.

Training Objective The parameters θ of the normalizing flow are optimized by maximizing the likelihood of the observed data $\{\mathbf{x}_i\}_{i=1}^N$, which is equivalent to minimizing the negative log-likelihood:

$$\theta^* = \arg\min_{\theta} \sum_{i=1}^{N} -\log p_X^{\theta}(\mathbf{x}_i).$$

By leveraging the exact computation of the log-likelihood, normalizing flows can be trained using standard gradient-based optimization methods, ensuring stable and efficient convergence.

Strengths Normalizing flows offer several advantages. A primary strength is their ability to perform **exact density computation**, allowing for tractable and exact likelihood evaluations. This facilitates stable training via maximum likelihood estimation [44]. Furthermore, they provide **efficient sampling**; once trained, new samples can be generated rapidly by transforming latent samples through G^{θ}. The analytical invertibility of G^{θ} and efficient computation of Jacobian determinants make both training and sampling computationally efficient.

Weaknesses Despite their strengths, normalizing flows have limitations. The requirement for invertibility and tractable Jacobians imposes constraints on the choice of transformations, potentially leading to **limited expressivity** [44]. This can hinder the model's ability to capture complex data distributions, especially those with intricate structures. Additionally, normalizing flows can encounter **topological limitations and numerical stability issues** when modeling targets with complicated topologies. In such cases, the model may become numerically non-invertible to approximate the target distribution closely, leading to stability problems and reducing the effectiveness of the flow [6, 11].

Auto-Regressive Models

Auto-regressive models are a class of generative models that factorize the joint probability distribution of high-dimensional data into a product of conditional distributions, leveraging the chain rule of probability. Notable examples include PixelCNN and PixelRNN [41], WaveNet [40], and more recently, Transformer-based models [64] applied across various domains such as natural language processing, image generation, and audio synthesis.

Model Formulation Given a data vector $\mathbf{x} = [x_1, x_2, \dots, x_d]$ in $\mathbb{R}^d$, the joint distribution $P(\mathbf{x})$ is decomposed into a product of conditional distributions:

$$P(\mathbf{x}) = \prod_{i=1}^{d} P(x_i | x_{1:i-1})$$

where $x_{1:i-1}$ denotes all preceding variables in a predefined ordering, which could be temporal (e.g., in time series data) or spatial (e.g., pixels in an image). This factorization allows the model to capture complex dependencies between variables by modeling each variable conditioned on all previous ones.

Modeling Conditional Distributions Each conditional distribution $P(x_i|x_{1:i-1})$ is modeled using a neural network that takes $x_{1:i-1}$ as input and outputs the parameters of the distribution for x_i. For instance, in PixelCNN, convolutional neural networks with masked filters are employed to ensure that the prediction of a pixel x_i depends only on pixels $x_{1:i-1}$ that come before it in the chosen ordering.

The masking technique is critical; it zeroes out connections in the network to prevent information from "future" variables influencing the prediction of the current variable, thus maintaining the auto-regressive property.

Training Objective The parameters θ of the auto-regressive model are optimized by maximizing the likelihood of the observed data $\{\mathbf{x}_i\}_{i=1}^{N}$, which is equivalent to minimizing the negative log-likelihood:

$$\theta^* = \arg\min_{\theta} \sum_{i=1}^{N} -\log P_\theta(\mathbf{x}_i) = \arg\min_{\theta} \sum_{i=1}^{N} \sum_{j=1}^{d} -\log P_\theta(x_{i,j}|x_{i,1:j-1})$$

This objective function allows for exact computation of the log-likelihood, facilitating stable training using gradient-based optimization methods.

Sampling Process Sampling from auto-regressive models is inherently sequential. To generate a new data point $\mathbf{x}$, the model starts by sampling x_1 from $P(x_1)$. Then, for $i = 2$ to d, each x_i is sampled from $P(x_i|x_{1:i-1})$, using the previously sampled values. This process continues until all variables have been generated.

Strengths Auto-regressive models have several notable strengths. One of their primary advantages is their ability to perform **exact likelihood computation**, providing an exact calculation of the data likelihood without any approximation. This facilitates stable training via maximum likelihood estimation, ensuring that the model accurately captures the underlying data distribution. Additionally, by conditioning each variable on all previous ones, auto-regressive models are highly effective at **modeling complex dependencies** in sequential or structured data. This makes them particularly well-suited for tasks such as language modeling, image generation, and audio synthesis, where capturing intricate patterns and dependencies is crucial. Furthermore, auto-regressive models offer **flexibility in handling different data types**; they can model both discrete and continuous data by appropriately choosing the form of the conditional distributions, such as using categorical distributions for discrete variables or Gaussian distributions for continuous ones.

Weaknesses Despite these strengths, auto-regressive models also present significant weaknesses. A major limitation is the **slow sampling process** inherent in their sequential nature. Generating a single data sample requires sampling each variable one at a time, conditioned on the previously generated variables. This makes the sampling process computationally intensive and time-consuming, especially for high-dimensional data like high-resolution images or long sequences, where the number of variables d is large. Moreover, the sequential dependency of variables **limits parallelization during sampling**, as each variable depends on the outcome of the previous ones, preventing the use of parallel computation techniques to speed up sample generation. This can be particularly problematic when rapid generation of many samples is required.

Energy-Based Models (EBMs)

Energy-Based Models (EBMs) [35] define an unnormalized probability distribution over data using an energy function $E_\theta(\mathbf{x})$. The probability of a data point $\mathbf{x}$ is given as:

$$P_\theta(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}))}{Z_\theta}$$

where $Z_\theta = \int \exp(-E_\theta(\mathbf{x}))d\mathbf{x}$ is the partition function, a normalizing constant that ensures $P_\theta(\mathbf{x})$ integrates to 1. However, computing Z_θ is typically intractable for high-dimensional data, posing challenges for direct optimization and sampling.

Training Objective EBMs are trained to assign low energy values to data samples (positive samples) and high energy values to non-data samples (negative samples). A commonly used training objective is based on contrastive divergence [25], which minimizes the expected energy of positive samples while maximizing the expected energy of negative samples:

$$\mathscr{L}(\theta) = \mathbb{E}_{\mathbf{x}^+ \sim P_{\text{data}}}[E_\theta(\mathbf{x}^+)] - \mathbb{E}_{\mathbf{x}^- \sim P_\theta}[E_\theta(\mathbf{x}^-)]$$

Here, $\mathbf{x}^+$ are positive samples drawn from the data distribution P_{data}, while $\mathbf{x}^-$ are negative samples generated from the model distribution P_θ.

To obtain the negative samples $\mathbf{x}^-$, we employ *Markov Chain Monte Carlo (MCMC)* methods, such as Langevin Dynamics. Langevin Dynamics iteratively updates a sample $\mathbf{x}_t$ to move toward regions of high probability under $P_\theta(\mathbf{x})$:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta \nabla_{\mathbf{x}} E_\theta(\mathbf{x}_t) + \sqrt{2\eta}\,\varepsilon_t$$

where η is the step size, and $\varepsilon_t \sim \mathcal{N}(0, I)$ adds Gaussian noise for stochasticity. The process ensures that the negative samples explore the energy landscape effectively, although the iterative updates make it computationally expensive.

Sampling Process Sampling from EBMs follows a similar procedure as obtaining negative samples. Since direct sampling is infeasible due to the intractability of Z_θ, MCMC methods like Langevin Dynamics are used to approximate samples from the target distribution. This iterative process converges to high-density regions of $P_\theta(\mathbf{x})$, making it suitable for generation tasks. However, the need for multiple iterations results in a slow sampling process, particularly for high-dimensional data.

Strengths EBMs offer several notable strengths. One of their primary advantages is their **flexibility** in representing complex distributions without requiring explicit normalization, as they define an unnormalized probability distribution directly using an energy function [35]. This flexibility allows EBMs to focus on modeling the energy landscape of the data, making them highly adaptable to diverse data types and problem settings. Unlike other generative models, such as GANs or normalizing flows, which often require specific assumptions about the form of the probability distribution or rely on explicit normalization, EBMs can represent distributions of arbitrary complexity by appropriately designing the energy function.

Furthermore, EBMs exhibit strong **compositionality**, allowing seamless integration with other models and the incorporation of domain-specific knowledge such as constraints or priors [14, 21]. This property enables EBMs to handle applications requiring hybrid generative processes or strict adherence to domain rules. By encoding physical laws or chemical properties directly into the energy function, EBMs can generate outcomes that comply with these constraints, making them particularly effective in scientific modeling tasks like physical simulations and molecular design [14]. In structured prediction tasks, EBMs can impose priors to ensure outputs follow specific patterns or relationships, such as spatial coherence in semantic segmentation or hierarchical structures in knowledge graph synthesis [21, 35]. This compositional and flexible nature allows EBMs to excel in applications requiring constrained generative modeling or the integration of domain knowledge, where conventional generative models may struggle.

Weaknesses Despite their strengths, EBMs also face significant weaknesses. A major challenge is their **training difficulty**, primarily due to the reliance on MCMC methods for generating negative samples. MCMC sampling is computationally intensive, can suffer from slow convergence, and often requires careful tuning to ensure effective exploration of the energy landscape. Additionally, EBMs exhibit **slow sampling speed**, as generating a single sample requires running iterative Langevin Dynamics or similar methods, which involve multiple gradient evaluations. This can hinder the scalability of EBMs for tasks requiring large-scale generation.

Diffusion and Score-Based Models

Diffusion models [26] and score-based generative models (SGMs) [61] are powerful generative frameworks that treat data generation as the process of reversing noise corruption. These models consist of two complementary components: a forward process that gradually perturbs data into a noise-like prior distribution and a reverse process that reconstructs the data by inverting this transformation. The forward process maps a complex data distribution into a simpler, analytically tractable distribution, typically the standard normal distribution $\mathcal{N}(0,\mathbf{I})$, while the reverse process transforms noise back into samples from the original data distribution.

In discrete-time models, such as Denoising Diffusion Probabilistic Models (DDPMs), this transformation occurs over a finite sequence of time steps, governed by a Markov chain. In contrast, continuous-time frameworks, such as Score-Based Generative Models (SGMs), model this progression using stochastic differential equations (SDEs). As we will see, these two frameworks are mathematically equivalent in the limit of infinite time steps. The reverse process, which lies at the heart of both frameworks, is the critical mechanism for generating high-quality samples.

Forward Process The forward process in diffusion and score-based generative models progressively corrupts the original data into noise, providing the foundation for learning the reverse process that generates data from noise. The design of the forward process ensures that over time, the data distribution transitions smoothly into a simple, analytically tractable distribution π, such as $\mathcal{N}(0,\mathbf{I})$.

- **DDPM (Discrete Time):**

In the Denoising Diffusion Probabilistic Model (DDPM), the forward process is a Markov chain that adds Gaussian noise to the data at discrete time steps $t = 1, 2, \ldots, N$. The transition kernel is given by:

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{\alpha_t}\,\mathbf{x}_{t-1}, \beta_t \mathbf{I}\right),$$

where:

- $\alpha_t = 1 - \beta_t$, the scaling factor applied to $\mathbf{x}_{t-1}$,
- $\beta_t \in (0,1)$, the predefined variance schedule controlling the amount of noise at step t,

This formulation allows the process to be expressed in terms of the initial data $\mathbf{x}_0$:

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0, (1 - \bar{\alpha}_t)\mathbf{I}\right),$$

where $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$ is the cumulative product of α_s. This closed-form expression simplifies training and allows direct sampling from any step t.

- **SGM (Continuous Time):** The forward process in Score-Based Generative Models (SGM) is described by a continuous-time Stochastic Differential Equation (SDE) that progressively perturbs the data distribution with infinitesimal noise:

$$d\mathbf{x} = f(\mathbf{x}, t)\,dt + g(t)\,d\mathbf{w},$$

where:

- $\mathbf{w}(t)$ is a standard Wiener process (Brownian motion),
- $f(\mathbf{x}, t)$ is the drift coefficient, defining the deterministic evolution of the data distribution,
- $g(t)$ is the diffusion coefficient, controlling the scale of the stochastic noise.

The drift and diffusion coefficients $f(\mathbf{x}, t)$ and $g(t)$ are carefully chosen so that, after a sufficiently large diffusion time T, the data distribution evolves into a distribution which is very close to a simple, analytically tractable distribution, typically the standard normal distribution $\mathcal{N}(0, \mathbf{I})$.

Popular choices for the forward SDE include:

- **Variance Preserving (VP) SDE:**

  $$f(\mathbf{x},t) = -\frac{1}{2}\beta(t)\mathbf{x}, \quad g(t) = \sqrt{\beta(t)},$$

  where $\beta(t)$ is a predefined noise schedule that controls the rate of diffusion. This choice preserves the variance of the data distribution throughout the process. The limiting distribution as $t \to \infty$ is the standard normal distribution. In practice, after sufficient diffusion time T, the perturbed distribution $p_T \approx \mathcal{N}(0,\mathbf{I})$.

- **Variance Exploding (VE) SDE:**

  $$f(\mathbf{x},t) = \mathbf{0}, \quad g(t) = \sqrt{\frac{d[\sigma^2(t)]}{dt}},$$

  where $\sigma(t)$ is a monotonically increasing function that scales the noise over time. In this formulation, the variance of the data distribution "explodes". After sufficient diffusion time T, the perturbed distribution $p_T \approx \mathcal{N}(0,\sigma(T)^2\mathbf{I})$.

These formulations ensure a smooth progression from the data distribution to the prior, providing a well-defined framework for the reverse process to reconstruct data from noise.

Reverse Process The reverse process reconstructs the original data distribution by inverting the forward process, progressively removing noise to generate data samples:

- **DDPM (Discrete Time):** The reverse process in DDPMs is a Markov chain that denoises the data at each step. The conditional distribution for each step is given by:

  $$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}\left(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t,t), \beta_t^2\mathbf{I}\right),$$

  where $\mu_\theta(\mathbf{x}_t,t)$ is a neural network predicting the mean of the denoised distribution based on the noisy input $\mathbf{x}_t$ and the timestep t. Sampling proceeds iteratively by drawing samples from $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$, starting from $\mathbf{x}_T \sim \pi = \mathcal{N}(0,\mathbf{I})$.

- **SGM (Continuous Time):** In SGMs, the reverse process is governed by the reverse-time SDE, derived from Anderson's theorem [1]:

  $$d\mathbf{x} = \left[f(\mathbf{x},t) - g(t)^2\nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right] dt + g(t)\, d\bar{\mathbf{w}},$$

  where:

- $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ is the gradient of the logdensity (score function) of the perturbed distribution at diffusion time t. In practice, the score function is approximated by a neural network $s_\theta(\mathbf{x},t)$, which we call the score model,
- $\bar{\mathbf{w}}(t)$ is a reverse-time Wiener process.

Sampling proceeds by solving this reverse-time SDE, starting from $\mathbf{x}_T \sim \pi \approx p_T(\mathbf{x})$ (e.g., $\mathcal{N}(0,\mathbf{I})$) and integrating backward from $t = T$ to $t = 0$. This iterative process generates high-quality samples that closely follow the original data distribution.

The reverse process lies at the core of diffusion-based generative modeling, enabling the transformation of simple noise into complex data distributions while preserving high fidelity.

Training Objective Training involves learning to approximate the reverse process by minimizing specific loss functions:

- **DDPM:** In the DDPM framework, the training objective is formulated to minimize an expression that serves as an upper bound on the negative log-likelihood of the data. This objective is defined as:

$$\mathcal{L}_{\text{DDPM}}(\theta) = \sum_{t=1}^{T} \mathbb{E}_{\mathbf{x}_0,\varepsilon_t}\left[\left\|\varepsilon_t - \varepsilon_\theta\left(\sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\varepsilon_t, t\right)\right\|^2\right],$$

  where:

  - $\varepsilon_t \sim \mathcal{N}(0,\mathbf{I})$ is Gaussian noise added at time step t,
  - $\varepsilon_\theta(\mathbf{x}_t,t)$ is a neural network that estimates the noise component in $\mathbf{x}_t$,
  - $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$ is the cumulative product of the noise schedule parameters,
  - $\alpha_t = 1 - \beta_t$, and β_t is the variance of the noise added at each time step.

  By minimizing $\mathcal{L}_{\text{DDPM}}(\theta)$, we effectively minimize an upper bound on the negative log-likelihood, enhancing the model's capacity to generate realistic data samples.

  The mean μ_θ of the reverse process used during sampling is derived from the predicted noise ε_θ as follows:

$$\mu_\theta(\mathbf{x}_t,t) = \frac{\sqrt{\bar{\alpha}_{t-1}}\,\beta_t}{1-\bar{\alpha}_t}\hat{\mathbf{x}}_0 + \frac{\sqrt{\alpha_t}\,(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t,$$

  where:

- $\hat{\mathbf{x}}_0 = \frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\,\varepsilon_\theta(\mathbf{x}_t,t)}{\sqrt{\bar{\alpha}_t}}$ is the reconstructed noiseless data point.

This formulation ensures that the reverse diffusion process effectively removes noise at each step, gradually transforming pure noise into high-quality data samples during generation.

- **SGM:** In the Score-Based Generative Modeling (SGM) framework, the neural network s_θ is trained to approximate the score function of the perturbed data distribution $p_t(\mathbf{x})$. This is achieved by minimizing the **Weighted Denoising Score Matching Objective**:

$$\mathscr{L}_{\text{DSM}}(\theta) = \frac{1}{2}\int_0^T \lambda(t)\,\mathbb{E}_{\mathbf{x}_0,\mathbf{x}_t}\left[\|s_\theta(\mathbf{x}_t,t) - \nabla_{\mathbf{x}_t}\log p_{0t}(\mathbf{x}_t \mid \mathbf{x}_0)\|^2\right] dt,$$

where:

  - $p_{0t}(\mathbf{x}_t \mid \mathbf{x}_0)$ is the perturbation kernel of the forward Stochastic Differential Equation (SDE),
  - $\lambda(t)$ is a weighting function,
  - $\mathbf{x}_t$ is the diffused data at time t.

[59] showed that for the particular choice $\lambda(t) = g(t)^2$, where $g(t)$ is the diffusion coefficient of the forward SDE, the $\mathscr{L}_{\text{DSM}}$ objective serves as an upper bound on the negative log-likelihood.

In practice, a different weighting function—often referred to as the *simple weighting*—is commonly used:

$$\lambda(t) = \sigma(t)^2,$$

where $\sigma(t)^2$ denotes the variance of the forward perturbation kernel $p_{0t}(\mathbf{x}_t \mid \mathbf{x}_0)$. This weighting function has been found to yield superior generative performance, particularly in the image domain where it was initially tested. As a result, it has become the *de facto* standard choice for the weighting function in practical implementations.

Equivalence Between DDPM and SGM Denoising Diffusion Probabilistic Models (DDPMs) and Score-Based Generative Models (SGMs) are equivalent in the limit as the number of timesteps $N \to \infty$. Specifically, the forward process in a DDPM converges to the Variance Preserving Stochastic Differential Equation (VP-SDE) used in SGMs:

$$d\mathbf{x} = -\tfrac{1}{2}\beta(t)\mathbf{x}\,dt + \sqrt{\beta(t)}\,d\mathbf{w},$$

where $\beta(t)$ is the continuous noise schedule and $\mathbf{w}$ is a standard Wiener process. This result is detailed in Song et al.'s paper *Score-Based Generative Modeling through Stochastic Differential Equations* (2020).

Probability Flow ODE

An alternative to the stochastic reverse-time SDE is the **probability flow ODE**, which provides a deterministic mapping between the data distribution and the prior. The probability flow ODE is defined as:

$$\frac{d\mathbf{x}}{dt} = f(\mathbf{x},t) - \frac{1}{2}g(t)^2 s_\theta(\mathbf{x},t),$$

where:

- $f(\mathbf{x},t)$ and $g(t)$ are the drift and diffusion coefficients of the forward SDE,
- $s_\theta(\mathbf{x},t)$ is the approximation of the score function at time t.

Under the assumption of a perfectly approximated score function, the probability flow ODE shares the same marginal probability distributions as the reverse-time SDE for all diffusion times. This equivalence arises from considering the Fokker-Planck equations corresponding to both processes, which govern the time evolution of probability densities.

Computational Efficiency. In practice, integrating the probability flow ODE has been observed to be computationally more efficient than simulating the reverse-time SDE. Fewer integration steps are typically required to achieve similar sample quality, making it a preferred choice for sampling in diffusion models. This efficiency stems from the deterministic nature of the ODE, which allows for larger step sizes, unlike the stochastic SDE that often requires smaller step sizes to maintain sample fidelity.

Likelihood Computation with the Probability Flow ODE. The probability flow ODE enables likelihood computation via the instantaneous change of variables formula. Given the score function $s_\theta(\mathbf{x},t)$, the log-likelihood of a data point $\mathbf{x}_0$ can be expressed as:

$$\log p_0(\mathbf{x}_0) = \log p_T(\mathbf{x}_T) + \int_0^T \nabla_{\mathbf{x}} \cdot \mathbf{f}_\theta^{\mathrm{ODE}}(\mathbf{x},t)\, dt,$$

where $\mathbf{f}_\theta^{\mathrm{ODE}}(\mathbf{x},t)$ is the drift term of the probability flow ODE. Direct computation of the divergence term $\nabla_{\mathbf{x}} \cdot \mathbf{f}_\theta^{\mathrm{ODE}}(\mathbf{x},t)$ is often computationally expensive. To address this, the Skilling-Hutchinson trace estimator is used:

$$\nabla_{\mathbf{x}} \cdot \mathbf{f}_\theta^{\mathrm{ODE}}(\mathbf{x},t) \approx \mathbb{E}_{\varepsilon\sim\mathcal{N}(0,\mathbf{I})}\left[\varepsilon^\top \nabla_{\mathbf{x}} \mathbf{f}_\theta^{\mathrm{ODE}}(\mathbf{x},t)\varepsilon\right],$$

where $\nabla_{\mathbf{x}}\mathbf{f}_{\theta}^{\text{ODE}}(\mathbf{x},t)$ is the Jacobian of $\mathbf{f}_{\theta}^{\text{ODE}}$. The term inside the expectation can be efficiently computed using Jacobian-vector product calculations, making this approach both practical and scalable for accurate likelihood computation.

Strengths and Weaknesses of the Unified Framework The unified framework of diffusion and score-based generative models exhibits several notable strengths. Among its strengths, the framework ensures stable training by avoiding issues such as mode collapse and instability that commonly afflict adversarial training methods like Generative Adversarial Networks (GANs). Its flexibility allows it to model complex, high-dimensional, and multi-modal data distributions, making it highly adaptable for various applications. Additionally, it is grounded in stochastic process theory with well-defined objectives, providing a solid theoretical foundation that guides both its development and analysis. However, a significant weakness of this framework is its slow sampling speed. Generating samples typically requires hundreds or even thousands of iterative steps, resulting in significant computational costs. This can make the approach impractical for applications where rapid sample generation is crucial.

Advancements in Sampling Speed Significant progress has been made in accelerating the sampling speed of diffusion models, effectively addressing their primary limitation of slow generation due to iterative steps. A critical observation in this area is that the probability flow ODE, which shares the same marginal distribution as the reverse SDE, is generally preferred for faster sampling. The ODE formulation tends to produce better samples than the corresponding SDE in the low discretization regime [60]. Consequently, many methods that accelerate diffusion models focus on learning to efficiently integrate the probability flow ODE, often as a post-training optimization.

Techniques such as **Denoising Diffusion Implicit Models (DDIM)** [57] reformulate the reverse diffusion process by utilizing non-Markovian dynamics, significantly reducing the number of required sampling steps without compromising sample quality. **Latent Diffusion Models (LDMs)** [51] enhance efficiency by performing diffusion in a compressed latent space, drastically decreasing computational demands while preserving high-resolution outputs. **Progressive Distillation** [53] further compresses the multi-step diffusion process into models capable of generating high-fidelity samples in significantly fewer steps—sometimes even a single step. Recently, **Consistency Models** [58] have been introduced, directly mapping noise to data in one step while retaining the advantages of iterative methods.

Collectively, these advancements have significantly reduced sampling times and made diffusion models practical for real-time and interactive applications across various domains.

1.2 Self-Supervised Learning

1.2.1 Motivation

In today's data-rich environment—spanning social media, sensor outputs, and extensive collections of images, videos and text—the primary challenge is not the lack of data but our ability to utilize it effectively. Despite the vast amounts of information available, much remains unlabeled and underexploited due to the high costs and impracticalities associated with manual labeling. This situation poses a crucial question: how can we unlock the potential of unlabeled data to develop intelligent systems that learn, adapt, and generalize in a manner akin to human learning?

Self-supervised learning (SSL) emerges as a compelling approach to this challenge. Referred to by Yann LeCun as the "dark matter of intelligence" [32], SSL enables machines to learn from the abundant unlabeled data by exploiting the inherent structures within it. LeCun remarks:

> "Common sense helps people learn new skills without requiring massive amounts of teaching for every single task. Self-supervised learning is one of the most promising ways to build such background knowledge and approximate a form of common sense in AI systems."

Analogous to how a child learns about the world through observation and pattern recognition rather than explicit instruction, SSL allows models to derive meaningful representations from data without manual labels. The transformative impact of SSL is evident across multiple domains::

Natural Language Processing (NLP): SSL has transformed NLP by enabling models to understand and generate human language with improved contextual awareness. Early models like Word2Vec [37] and GloVe [45] learned semantic relationships between words through context prediction. Advanced models such as BERT [12] and GPT [48] employed masked language modeling to pretrain on extensive text corpora, enhancing capabilities in tasks like text completion and machine translation.

Computer Vision (CV): In CV, SSL techniques like Contrastive Predictive Coding (CPC) [42], SimCLR [9], and MoCo [24] have enabled models to learn visual representations without reliance on labeled datasets. For example, Facebook's SEER model [20], trained on a billion unlabeled images, achieved state-of-the-art performance on ImageNet, illustrating the efficacy of learning from unlabeled visual data.

Speech Recognition: SSL models such as wav2vec [54] and wav2vec 2.0 [3] have significantly advanced speech recognition by learning representations from raw audio data.

By predicting future audio samples from past ones, these models capture essential phonetic and linguistic features, improving transcription accuracy and the handling of diverse accents and intonations.

Multimodal Learning: SSL has significantly advanced the integration of multiple data modalities, enabling AI systems to learn joint representations that bridge the gap between domains like vision, language, and audio. Pioneering models such as CLIP [47] have shown remarkable success in aligning images and text within a shared latent space, enabling a wide range of applications. These include image captioning, where the model generates descriptive text for images; text-based image retrieval, allowing users to find images using natural language queries; and zero-shot classification, where the model classifies images into categories it has not seen during training, guided only by textual descriptions.

These advancements are not merely incremental; they signify a fundamental shift in machine learning methodologies. By leveraging unlabeled data, SSL enables models to develop a form of common sense, understanding context and patterns previously inaccessible through supervised learning alone.

The potential applications of SSL are extensive. It opens pathways to developing AI systems capable of understanding and generating human-like language and vision, and learning new tasks with minimal supervision. Self-supervised learning represents a significant step toward creating AI that learns and adapts more like humans, enhancing intelligence and adaptability in artificial systems.

1.2.2 Mathematical Formulation of SSL

Formally, let $\mathscr{D} = \{\mathbf{x}_i\}_{i=1}^{N} \subset \mathbb{R}^d$ be a dataset of unlabeled samples drawn from an unknown distribution $P_{\mathbf{x}}$. The objective of self-supervised representation learning is to learn an encoder function $f_\theta : \mathbb{R}^d \to \mathbb{R}^k$, where $k \leq d$, that maps input data to a latent space, producing meaningful representations $\mathbf{h}_i = f_\theta(\mathbf{x}_i)$.

The encoder f_θ is parameterized by θ and is trained by minimizing a self-supervised loss function $\mathscr{L}(\theta; \mathscr{D})$ designed to capture essential features of the data:

$$\theta^* = \arg\min_\theta \mathbb{E}_{\mathbf{x}_i \sim P_{\mathbf{x}}}[\mathscr{L}(\mathbf{x}_i; \theta)].$$

The design of $\mathscr{L}$ depends on the specific self-supervised learning approach and aims to enforce properties such as invariance to data augmentations, ability to predict contextual information, or alignment with semantic structures.

1.2.3 Generative vs. Discriminative SSL

Self-supervised learning methods can be broadly categorized into two paradigms: **generative** and **discriminative** approaches. Understanding the strengths and applications of each is crucial for selecting appropriate methods for specific tasks.

Generative Approaches

These methods aim to extract meaningful latent representations by modeling the underlying data distribution. These approaches can be broadly categorized into two types based on the scope of the distribution being modeled:

1. **Full data distribution** $p(\mathbf{x})$
2. **Maked data distribution** $p(\mathbf{x}_{\mathbf{masked}}|\mathbf{x}_{\mathbf{context}})$

Learning Latent Representations by Modeling the Full Distribution $p(\mathbf{x})$

- *Auto-Regressive (AR) Models*: AR models factorize the data distribution into a product of conditional probabilities. These models generate each data element sequentially, conditioned on the previous ones.
  - *GPT and GPT-2* [48]: Learn language representations by predicting the next token given preceding context, enabling robust text generation and understanding. GPT embeddings serve as inputs for fine-tuned classifiers in NLP tasks.
  - *PixelRNN and PixelCNN* [41]: Model pixel dependencies in images for self-supervised representation learning, useful for tasks like image completion. PixelCNN representations have been used in image segmentation.
- *Normalizing Flows*: These models transform a simple latent distribution into a complex target distribution using a sequence of invertible mappings, preserving exact likelihoods. Examples include:
  - *FlowGMM* [28]: Combines normalizing flows with a Gaussian mixture model in the latent space to extract representations useful for tasks such as text classification, tabular data analysis, and semi-supervised image classification.
- *Auto-Encoding (AE) Models*: Autoencoders reconstruct data from latent representations, learning robust embeddings for various downstream tasks. Examples include:

  - *Variational Autoencoders (VAEs)* [31]: Introduce probabilistic latent spaces, learning smooth representations useful for clustering and semi-supervised learning.
  - *VQ-VAE* [43]: Extends VAEs to discrete latent spaces, providing meaningful representations that have been widely applied in downstream tasks:
    - In *audio processing*, VQ-VAE representations have been used in *wav2vec* [3] for unsupervised pretraining on speech data, enabling improvements in downstream speech recognition tasks by learning phonetic and linguistic features from raw audio.
    - In *vision*, the latent codes are inputs to autoregressive models for image generation, inpainting, and enhancement tasks [49].
    - In *multimodal learning*, VQ-VAE representations bridge modalities such as text and images, aiding in tasks like image captioning and cross-modal retrieval [13].

- *GAN-Based Models*: Generative Adversarial Networks (GANs) learn latent representations by training a generator and a discriminator adversarially. Recent advancements adapt GANs for representation learning. Examples include:
  - *StyleGAN* [29]: Generates high-quality images with interpretable latent spaces that enable fine-grained editing (e.g., facial expressions and lighting adjustments). StyleGAN embeddings are widely used in image synthesis and manipulation tasks.
  - *VQ-GAN* [15]: Combines vector quantization and adversarial training to learn discrete latent representations. These representations are used for tasks such as text-to-image generation, image compression, and cross-modal retrieval.

Learning Latent Representations by Modeling the Masked Distribution $p(\mathbf{x}_{\textbf{masked}}|\mathbf{x}_{\textbf{context}})$

- **Masked Image Modeling:** Models learn representations by reconstructing occluded parts of an image, leveraging context to infer the missing information.
  - *Examples:*
    - **BEiT** [4]: Learns representations for downstream tasks like image classification and segmentation by reconstructing masked patches of images.
    - **Masked Autoencoders (MAE)** [23]: Focuses on learning efficient representations by masking a large portion of the input image and reconstructing it from the unmasked portion.

  - *Downstream Applications:* Representations from masked image models are particularly useful for visual tasks such as object detection, segmentation, and classification, where capturing spatial relationships is critical.

- **Masked Language Modeling:** In NLP, masked language models predict missing words or tokens in a sequence, learning rich contextual embeddings.

  - *Examples:*

    * **BERT** [12]: A foundational model in NLP, pretraining on masked tokens enables fine-tuning for tasks such as sentiment analysis and named entity recognition.
    * **RoBERTa** [36]: Builds on BERT by optimizing training dynamics and is widely used for text classification and question answering.

  - *Downstream Applications:* Masked language model representations are extensively used for text-based tasks, including translation, summarization, and retrieval.

Discriminative Approaches

These methods focus on learning robust feature representations by solving pretext tasks or leveraging contrastive objectives without explicitly modeling the data distribution. The main motivation is to learn representations that are useful for downstream tasks by capturing essential features and invariances in the data. Key techniques include:

- **Contrastive Learning**:

  - *Motivation and Logic*: Contrastive learning aims to learn robust feature representations by maximizing agreement between similar pairs (positive pairs) and minimizing agreement between dissimilar pairs (negative pairs). The central idea is to structure the latent space such that representations of similar samples are closer, while representations of different samples are farther apart.

  - *Generalized Contrastive Loss*: [63] unified contrastive losses under a general framework, where the loss is expressed as:

$$\mathscr{L}_{\phi,\psi}(\theta) = \sum_{i=1}^{N} \phi\left(\sum_{j\neq i} \psi\left(\|z_i - z_j\|_2^2 - \|z_i - z_i'\|_2^2\right)\right),$$

    where:

    * z_i and z'_i are the representations of a sample i and its positive pair (augmented view of the same sample).
    * $||z_i - z_j||_2^2$ is the squared distance between the representations of sample i and a negative sample j.
    * ϕ and ψ are monotonically increasing and differentiable scalar functions that control how positive and negative distances are weighted.

  Specific choices of ϕ and ψ lead to well-known contrastive learning methods:

    * **InfoNCE Loss** [42]: Setting $\phi(x) = \tau \log(\varepsilon + x)$ and $\psi(x) = \exp(x/\tau)$ results in:

$$\mathscr{L}_{\text{InfoNCE}} = -\tau \sum_{i=1}^{N} \log \frac{\exp\left(-||z_i - z'_i||_2^2/\tau\right)}{\varepsilon \exp\left(-||z_i - z'_i||_2^2/\tau\right) + \sum_{j \neq i} \exp\left(-||z_i - z_j||_2^2/\tau\right)}.$$

    * **SimCLR Loss** [9]: Simplifies InfoNCE by setting $\varepsilon = 0$, removing the stabilization term in the denominator.

- **Self-Distillation**:
  - *Motivation and Logic*: These methods aim to learn representations without the need for negative samples, addressing issues like the need for large batch sizes or memory banks in contrastive learning. The model learns by predicting its own representations under different augmentations, promoting consistency.
  - *Training Objective*: The model minimizes a prediction loss between representations of different augmented views. For instance, in BYOL, the loss is:

$$\mathscr{L}_{\text{BYOL}} = \left|\left|\mathbf{q}_i - \text{stopgrad}(\mathbf{z}_i^+)\right|\right|_2^2,$$

    where:

    * $\mathbf{q}_i = f_\theta(\mathbf{x}_i)$ is the output of the online network.
    * $\mathbf{z}_i^+ = f_{\theta'}(\mathbf{x}_i^+)$ is the output of the target network.
    * stopgrad indicates that gradients are not backpropagated through this path.
    * θ' is an exponential moving average of the parameters θ.

  - *Examples*:
    * *BYOL* [22]: Uses an online and a target network to predict one view from another, with the target network updated via an exponential moving average.
    * *SimSiam* [10]: Simplifies BYOL by removing the momentum encoder and employing a stop-gradient operation to prevent representational collapse.

- **Feature Decorrelation Methods**:
  - *Motivation and Logic*: Feature decorrelation methods aim to learn rich and diverse representations by reducing redundancy among feature dimensions. Inspired by Canonical Correlation Analysis (CCA), these approaches encourage each component of the embedding to capture unique information, promoting uncorrelated and informative features. By focusing on decorrelating feature dimensions, these methods prevent representational collapse and enhance the quality of learned representations without relying on negative samples or complex training schemes.
  - *Training Objective*: The central idea is to make embeddings from different augmented views of the same data similar (invariance) while ensuring that different feature dimensions are uncorrelated (decorrelation). This is achieved by combining an invariance term with a decorrelation term in the loss function, encouraging both similarity across views and diversity among features.
    - *Barlow Twins* [66] employs two identical networks to process different augmented views of the same image. It computes a cross-correlation matrix $\mathbf{C}$ between the embeddings from the two networks. The loss function encourages the diagonal elements of $\mathbf{C}$ to be close to one (promoting invariance) and the off-diagonal elements to be close to zero (reducing redundancy):

$$\mathscr{L}_{\mathrm{BT}} = \sum_{i=1}^{d} (1 - \mathbf{C}_{ii})^2 + \lambda \sum_{i \neq j} \mathbf{C}_{ij}^2.$$

      This approach enables the learning of features that are invariant to augmentations while ensuring each dimension captures distinct information.
    - *VICReg* [5] introduces a loss function with three components. The invariance term minimizes the mean squared error between embeddings from different views. The variance term ensures the standard deviation of each feature dimension exceeds a threshold to prevent collapse. The covariance term penalizes off-diagonal covariance elements to reduce redundancy. The loss is expressed as:

$$\mathscr{L}_{\mathrm{VICReg}} = \underbrace{\frac{1}{B} \sum_{b=1}^{B} \|\mathbf{z}_b^{(1)} - \mathbf{z}_b^{(2)}\|_2^2}_{\text{Invariance}} + \gamma \underbrace{\sum_{j=1}^{d} \max(0, s - \sigma(\mathbf{z})_j)}_{\text{Variance}} + \mu \underbrace{\sum_{i \neq j} \mathrm{Cov}(\mathbf{z})_{ij}^2}_{\text{Covariance}},$$

where γ and μ balance the terms, s is the variance threshold, and $\text{Cov}(\mathbf{z})_{ij}$ is the off-diagonal element of the covariance matrix of $\mathbf{z}$.

- **Clustering-Based Methods**:
  - *Motivation and Logic*: These approaches leverage unsupervised clustering to group similar data points, learning representations that capture group-level semantic structures. The model alternates between clustering the data in the representation space and updating the encoder to produce features that align with these clusters.
  - *Training Objective*: The encoder is trained to predict pseudo-labels derived from cluster assignments. The loss function typically involves cross-entropy between the predicted labels and the pseudo-labels:

$$\mathscr{L}_{\text{cluster}} = \sum_{i=1}^{N} \ell_{\text{CE}}(f_\theta(\mathbf{x}_i), y_i),$$

    where y_i is the cluster assignment for $\mathbf{x}_i$ obtained via a clustering algorithm (e.g., k-means), and ℓ_{CE} denotes the cross-entropy loss.
  - *Examples*:
    * *DeepCluster* [7]: Performs k-means clustering on the learned features and uses the cluster assignments as pseudo-labels for training.
    * *SwAV* [8]: SwAV uses siamese networks to generate embeddings for two augmented views of an image and aligns them with trainable prototypes. It computes cluster assignments online using the Sinkhorn-Knopp algorithm and minimizes cross-entropy between these assignments and predicted probabilities, enabling efficient training without pairwise comparisons.

- **Pretext Task-Based Methods**
  - *Motivation and Logic*: Pretext task-based methods aim to learn useful representations by training models on auxiliary tasks where the labels can be derived automatically from the data itself. These tasks, often unrelated to the downstream tasks, encourage the model to capture meaningful patterns and structures in the data that generalize well across applications. By solving these simple yet informative tasks, the model learns features that are transferable to various downstream tasks without requiring manual annotations.
  - *Examples*:

* **Rotation Prediction** [17]: The model learns representations by predicting the rotation angle applied to an image. Given an image rotated by one of four angles (0°, 90°, 180°, or 270°), the model classifies the rotation angle. This task encourages the model to understand spatial relationships and object structures within the image.
* **Jigsaw Puzzle Solving** [39]: The model is tasked with solving jigsaw puzzles by predicting the correct arrangement of shuffled image patches. This pretext task forces the model to learn contextual and semantic relationships between different parts of an image.
* **Temporal Ordering** [38]: For video data, the model learns representations by predicting the correct temporal order of video frames. This task helps the model capture temporal dependencies and motion dynamics in videos.

Choosing Between Generative and Discriminative SSL

The choice between generative and discriminative self-supervised learning methods depends on the specific objectives and requirements of the task. Each approach has its strengths and is more suitable under certain circumstances.

Generative SSL is more useful when: Generative SSL methods excel in scenarios that demand a fine-grained understanding of the data distribution. These methods capture detailed and nuanced information within the extracted representations, making them ideal for tasks that require precise and intricate information about the data. For instance, in *image editing and manipulation*, generative models such as StyleGAN [29, 30] enable sophisticated modifications by navigating the latent space. By solving optimization problems to move in specific directions within this space, attributes like facial expressions, styles, or lighting conditions can be altered, providing a level of fine-grained control that discriminative models typically cannot offer. Additionally, generative SSL is advantageous in applications like anomaly detection, where understanding the complete data distribution allows for the identification of outliers, and in cross-modal tasks such as text-to-image generation, where translating between different data modalities requires comprehensive distribution modeling.

Discriminative SSL is more useful when: Discriminative SSL methods are preferable when the primary goal is to learn efficient, task-relevant representations tailored for specific downstream applications. These methods optimize embeddings to effectively distinguish between different classes or instances, making them highly suitable for tasks such as classification and retrieval. For example, in image classification, discriminative models like

SimCLR [9] and MoCo [24] produce embeddings that enhance performance by focusing on feature discrimination. Furthermore, discriminative SSL is advantageous in environments with limited computational resources, as these methods often require fewer resources and are more scalable. They are also ideal for real-time or latency-sensitive applications, where fast inference and low latency are critical, such as online recommendation systems or real-time monitoring.

Blurring the Lines: Hybrid Approaches

The distinction between generative and discriminative SSL methods is increasingly becoming blurred, with hybrid approaches emerging that combine the strengths of both paradigms. These hybrid models leverage the fine-grained distribution modeling of generative methods alongside the efficient, task-specific representation learning of discriminative methods. Notable examples include BEiT [4] and iBOT [67], which integrate masked image modeling (a generative task) with self-distillation objectives (a discriminative task). By doing so, these models capture rich data representations while maintaining computational efficiency, effectively addressing a broader range of tasks.

1.2.4 Utilizing Learned Representations in Downstream Tasks

Once the representation function f_θ is learned, the representations $\mathbf{h}_i = f_\theta(\mathbf{x}_i)$ can be utilized in various downstream tasks. These representations capture essential features and structures in the data, making them valuable inputs for different applications.

Classification

In classification problems, the goal is to assign input data to one of several predefined categories. The learned representations $\mathbf{h}_i$ serve as informative features that can be used to train a classifier. Specifically, we can define a classifier $g_\phi : \mathbb{R}^k \to \mathbb{R}^C$, where C is the number of classes, and ϕ represents the parameters of the classifier (e.g., weights of a fully connected layer).

The classifier is trained on a labeled dataset $\{(\mathbf{x}_j, y_j)\}_{j=1}^M$, where $y_j \in \{1, 2, \ldots, C\}$ is the class label for sample $\mathbf{x}_j$. The training involves minimizing a supervised loss function, typically the cross-entropy loss:

$$\phi^* = \arg\min_\phi \frac{1}{M} \sum_{j=1}^{M} \ell_{\text{CE}}(g_\phi(\mathbf{h}_j), y_j),$$

where $\mathbf{h}_j = f_\theta(\mathbf{x}_j)$ and ℓ_{CE} denotes the cross-entropy loss function.

The representations $\mathbf{h}_i$ often capture high-level semantic features and are invariant to variations in the input data that are irrelevant to the classification task, such as changes in lighting, orientation, or background. This invariance leads to better generalization and robustness, improving classification performance even when labeled data is limited.

In practice, the representations can be utilized in two ways:

- **Fixed Feature Extraction**: The encoder f_θ is kept frozen, and only the classifier g_ϕ is trained on the labeled data. This approach is useful when computational resources are limited or when the labeled dataset is small.
- **Fine-Tuning**: Both the encoder f_θ and the classifier g_ϕ are trained (or fine-tuned) jointly on the labeled data. This allows the model to adapt the learned representations to the specific nuances of the downstream task, often leading to better performance.

Other Downstream Tasks

Similar logic can be applied to other downstream tasks such as object detection, semantic segmentation, instance segmentation, image captioning, and more. In these tasks, the extracted representations $\mathbf{h}_i$ serve as inputs to task-specific prediction heads. For example, in object detection, the representations can be used to extract feature maps that help in localizing and classifying objects within images. In semantic segmentation, the representations provide spatial and contextual information necessary for assigning class labels to each pixel in an image. For image captioning, the representations capture visual features that can be input to sequence models to generate descriptive captions for images.

In all these tasks, the pretrained encoder f_θ acts as a feature extractor, and the task-specific prediction heads are trained to perform the desired task using the representations as input. Similar to classification, the encoder f_θ can be kept fixed or fine-tuned along with the prediction heads. This flexibility allows the model to either leverage the general representations learned during self-supervised pretraining or adapt them to the specific nuances of the downstream task. This approach leverages the strengths of self-supervised learning to enhance performance across various domains and applications.

1.2.5 Challenges and Weaknesses in Current Methodologies

Despite significant advancements, current unsupervised representation learning methods face several challenges:

- **Limited Model Expressiveness**: Generative models like standard VAEs struggle to capture complex data distributions due to limited decoder capacity and restrictive assumptions (e.g., Gaussianity). This limits their applicability to high-dimensional and complex data.

- **Intrinsic Dimension Estimation**: Many methods lack mechanisms to estimate the intrinsic dimensionality of the data manifold, which is crucial for understanding the true complexity and structure of the data.
- **Geometric Understanding**: Existing methods often fail to exploit the geometry of the data manifold effectively. Without leveraging geometric insights, models may learn suboptimal representations that do not reflect the true relationships in the data.

1.2.6 Addressing Weaknesses Through Our Work

In this thesis, we propose novel approaches to address these challenges in self-supervised representation learning:

ScoreVAE We introduce *ScoreVAE*, a model that integrates variational inference with pre-trained diffusion models to overcome the limitations of traditional VAEs. By relaxing the Gaussian assumption in the reconstruction distribution and incorporating more expressive score-based models, ScoreVAE achieves higher model expressiveness, enabling it to capture complex data distributions and produce clearer reconstructions.

Intrinsic Dimension Estimation Using Diffusion Models We propose a novel framework that leverages diffusion models to estimate the intrinsic dimensionality of data. By analyzing the diffusion process and its interaction with the data manifold, our method captures the latent structure and dimensionality, offering a principled way to understand complex datasets.

Score-Based Pullback Riemannian Geometry Our work introduces a method to learn the geometry of data manifolds by adapting a score-based Riemannian metric. This approach enables dimensionality reduction and geometric understanding while preserving local manifold properties. By leveraging the pullback metric induced by the score function, we address intrinsic dimension estimation and effectively exploit the underlying geometry in representation learning.

These contributions aim to enhance the expressiveness of generative models, provide tools for intrinsic dimension estimation, and integrate geometric insights into self-supervised learning, thereby advancing the field toward more robust and intelligent systems.

References

[1] Anderson, B. D. (1982). Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326.

[2] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein GAN. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70, pages 214–223. PMLR.

[3] Baevski, A., Zhou, Y., Mohamed, A., and Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 12449–12460.

[4] Bao, H., Dong, L., and Wei, F. (2021). Beit: Bert pre-training of image transformers. In *International Conference on Learning Representations*.

[5] Bardes, A., Ponce, J., and LeCun, Y. (2022). Vicreg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations (ICLR)*.

[6] Behrmann, J., Vicol, P., Wang, K.-C., Grosse, R., and Jacobsen, J.-H. (2021). Understanding and mitigating exploding inverses in invertible neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 1792–1800. PMLR.

[7] Caron, M., Bojanowski, P., Joulin, A., and Douze, M. (2018). Deep clustering for unsupervised learning of visual features. In *European Conference on Computer Vision (ECCV)*, pages 132–149.

[8] Caron, M., Misra, I., Mairal, J., et al. (2020). Unsupervised learning of visual features by contrasting cluster assignments. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 9912–9924.

[9] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning (ICML)*, pages 1597–1607.

[10] Chen, X. and He, K. (2021). Exploring simple siamese representation learning. *arXiv preprint arXiv:2011.10566*.

[11] Cornish, R., Caterini, A., Deligiannidis, G., and Doucet, A. (2020). Relaxing bijectivity constraints with continuously indexed normalising flows. In *International conference on machine learning*, pages 2133–2143. PMLR.

[12] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[13] Ding, Y., Hu, H., Ge, Z., Wu, J., Zhang, H., and Ding, C. (2021). Vq-vae for multimodal image synthesis. *arXiv preprint arXiv:2103.02398*.

[14] Du, Y. and Mordatch, I. (2019). Implicit generation and modeling with energy-based models. In *Advances in Neural Information Processing Systems*, volume 32.

[15] Esser, P., Rombach, R., and Ommer, B. (2021). Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12873–12883.

[16] Fefferman, C., Mitter, S., and Narayanan, H. (2016). Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29(4):983–1049.

[17] Gidaris, S., Singh, P., and Komodakis, N. (2018). Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations (ICLR)*.

[18] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT Press.

[19] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks.

[20] Goyal, P., Mahajan, D., Gupta, A., and Misra, I. (2021). Self-supervised pretraining of visual features in the wild. *arXiv preprint arXiv:2103.01988*.

[21] Grathwohl, W., Wang, K. C., Jacobsen, J.-H., Duvenaud, D., Swersky, K., and Norouzi, M. (2020). Your classifier is secretly an energy-based model and you should treat it like one. In *International Conference on Learning Representations (ICLR)*.

[22] Grill, J.-B., Strub, F., Altché, F., et al. (2020). Bootstrap your own latent: A new approach to self-supervised learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 21271–21284.

[23] He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. (2022). Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009.

[24] He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9729–9738.

[25] Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800.

[26] Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models.

[27] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

[28] Izmailov, P., Kirichenko, P., Finzi, M., and Wilson, A. G. (2021). Semi-supervised learning with normalizing flows. In *International Conference on Learning Representations*.

[29] Karras, T., Laine, S., and Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410.

[30] Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. (2020). Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8110–8119.

[31] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

[32] LeCun, Y. (2021). Self-supervised learning: The dark matter of intelligence. *Facebook Research*.

[33] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

[34] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[35] LeCun, Y., Chopra, S., and Hadsell, R. (2006). A tutorial on energy-based learning. *Predicting Structured Data*.

[36] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

[37] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. In *International Conference on Learning Representations (ICLR)*.

[38] Misra, I., Zitnick, C. L., and Hebert, M. (2016). Shuffle and learn: unsupervised learning using temporal order verification. In *European Conference on Computer Vision (ECCV)*, pages 527–544. Springer.

[39] Noroozi, M. and Favaro, P. (2016). Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision (ECCV)*, pages 69–84. Springer.

[40] Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016a). Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

[41] Oord, A. v. d., Kalchbrenner, N., and Kavukcuoglu, K. (2016b). Pixel recurrent neural networks. In *International Conference on Machine Learning (ICML)*, pages 1747–1756.

[42] Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 10256–10265.

[43] Oord, A. v. d., Vinyals, O., and Kavukcuoglu, K. (2017). Neural discrete representation learning. *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6306–6315.

[44] Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2021). Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64.

[45] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

[46] Poggio, T., Mhaskar, H., Rosasco, L., Miranda, B., and Liao, Q. (2017). Why and when can deep–but not shallow–networks avoid the curse of dimensionality: A review. *International Journal of Automation and Computing*, 14(5):503–519.

[47] Radford, A., Kim, J. W., Hallacy, C., et al. (2021). Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020.*

[48] Radford, A., Wu, J., Amodei, D., et al. (2019). Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

[49] Razavi, A., van den Oord, A., and Vinyals, O. (2019). Generating diverse high-fidelity images with vq-vae-2. In *Advances in Neural Information Processing Systems*, pages 14866–14876.

[50] Rezende, D. and Mohamed, S. (2015). Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538. PMLR.

[51] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695.

[52] Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326.

[53] Salimans, T. and Ho, J. (2022). Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512.*

[54] Schneider, S., Baevski, A., Collobert, R., Auli, M., and Mohamed, A. (2019). wav2vec: Unsupervised pre-training for speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6415–6419.

[55] Scott, D. W. (2015). *Multivariate Density Estimation: Theory, Practice, and Visualization.* John Wiley & Sons.

[56] Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis.* Chapman and Hall.

[57] Song, J., Meng, C., and Ermon, S. (2020a). Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502.*

[58] Song, Y., Dhariwal, P., Chen, M., and Sutskever, I. (2023). Consistency models. *arXiv preprint arXiv:2303.01469*.

[59] Song, Y., Durkan, C., Murray, I., and Ermon, S. (2021a). Maximum likelihood training of score-based diffusion models.

[60] Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2020b). Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*.

[61] Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2021b). Score-based generative modeling through stochastic differential equations.

[62] Tenenbaum, J. B., Silva, V. d., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323.

[63] Tian, Y. et al. (2022). Understanding self-supervised contrastive learning with generalized contrastive losses. *arXiv preprint arXiv:2202.09671*.

[64] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

[65] Wasserman, L. (2006). *All of Nonparametric Statistics*. Springer Science & Business Media.

[66] Zbontar, J., Jing, L., Misra, I., LeCun, Y., and Deny, S. (2021). Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning (ICML)*. PMLR.

[67] Zhou, D., Wang, Z., Wang, L., Han, J., Dai, X., Shen, Y., and Feng, J. (2022). ibot: Image bert pre-training with online tokenizer. In *International Conference on Learning Representations*.