

---

# Diffusion-Based Generative Models with Learned Anisotropic Covariance in 2D Space

---

**Keyi Chen**

Advisor: Rayan Saab  
Department of Mathematics  
University of California San Diego

## Abstract

Diffusion probabilistic models generate data by reversing a stochastic process that gradually transforms structured data into Gaussian noise. While effective, most existing models assume fixed isotropic or diagonal noise covariances, which limits their capacity to capture complex dependencies in the data. In this thesis, we consider a modified diffusion framework that learns a full anisotropic, input- and time-dependent covariance matrix in the reverse process. We provide a detailed mathematical derivation of the forward process, closed-form posteriors, and propose learning objectives that combine noise prediction with KL-divergence-based covariance prediction. The resulting model jointly learns both noise and covariance components. Experiments on 2D synthetic datasets (Swiss Roll, S-Curve, Moons, Circles) demonstrate improved sample fidelity, better reconstruction of geometric structures, and enhanced class separation. We further analyze noise scaling effects during sampling, illustrating trade-offs between diversity and structural sharpness. Our results suggest that full anisotropic covariance learning provides a more expressive and flexible extension to standard diffusion models.

## 1 Introduction

Generative models have become a popular topic in machine learning, with the goal of learning complex probability distributions from data and generating new samples that resemble the observed data. Among these, diffusion probabilistic models have emerged as a powerful framework, drawing inspiration from the physical process of diffusion — where data is progressively perturbed by random noise and subsequently reconstructed through a learned denoising procedure. These models define a forward stochastic process that incrementally adds noise to the data over a sequence of timesteps, along with a reverse process that learns to recover the original data distribution. At the core of the reverse process is a neural network trained to approximate the inverse dynamics of the diffusion process.

This thesis investigates the mathematical structure of diffusion models, with particular emphasis on the role of the noise covariance in the reverse process. Rather than assuming isotropic or diagonal Gaussian noise, we consider a more expressive model in which a neural network predicts a full, time- and input-dependent covariance matrix at each step of the denoising process. This extension introduces new mathematical complexity into the model, providing an opportunity to study the behavior of learned uncertainty and the geometry of the data distribution under stochastic refinement.

To explore these ideas in a controlled and mathematically interpretable setting, we conduct experiments on synthetic two-dimensional datasets that exhibit diverse geometric structures, including the Swiss roll, S-curve, moons, and concentric circles. These datasets capture nonlinear features such as curvature, branching, and cyclic symmetry, while remaining computationally tractable for repeated training and detailed analysis. In our experiments, we investigate the effect of learning a full, input- and time-dependent covariance matrix  $\Sigma_\theta(x_t, t)$  during the reverse denoising process, and evaluate

how this flexibility improves the model’s ability to adapt the noise structure and generate high-quality samples. Furthermore, we examine the role of class label conditioning by comparing models trained with and without access to shape category labels, assessing how conditional information influences reconstruction fidelity. Finally, we analyze the trade-off introduced by scaling the injected noise during sampling, demonstrating how varying the noise weight  $\lambda$  controls the balance between sample sharpness and diversity.

## 2 Background

### 2.1 Basics of Neural Networks

Mathematically, neural networks are functions from  $\mathbb{R}^{N_0}$  to  $\mathbb{R}^{N_L}$  that are parameterized by their weights. A common neural network structure is the feed-forward neural network which can be represented as

$$\Phi(x) = \rho \circ A^{(L)} \circ \rho \circ A^{(L-1)} \dots \circ \rho \circ A^{(1)}(x)$$

where  $\rho : \mathbb{R} \rightarrow \mathbb{R}$  is a nonlinear function applied elementwise, and each  $A^{(i)} : \mathbb{R}^{N_{i-1}} \rightarrow \mathbb{R}^{N_i}$  is an affine map defined by

$$A^{(i)}(x) = W^{(i)}x + b^{(i)}.$$

Here,  $W^{(i)} \in \mathbb{R}^{N_i \times N_{i-1}}$  are called weight matrices and  $b^{(i)} \in \mathbb{R}^{N_i}$  are the bias vectors. In particular, the rows of  $W^{(i)}$  are often referred to as neurons, and the composition  $\rho \circ A^{(i)}$  is typically called a layer in the network.

### 2.2 Training and Optimization

The parameters  $\{W^{(i)}, b^{(i)}\}$  are learned from data by minimizing a loss function that quantifies the discrepancy between the network’s output and the expected target. Specifically, consider a training dataset consisting of  $M$  input-output pairs, denoted by  $\{(x_j, y_j)\}_{j=1}^M$ , where each  $x_j \in \mathbb{R}^{N_0}$  is an input vector (e.g., a data point or feature vector), and  $y_j \in \mathbb{R}^{N_L}$  is the corresponding target output (e.g., a class label or regression value).

The learning objective is to find parameter values such that the network output  $\Phi(x_j)$  is close to the target  $y_j$  for all  $j$ . This is typically formulated as an optimization problem:

$$\min_{\{W^{(i)}, b^{(i)}\}} \frac{1}{M} \sum_{j=1}^M \mathcal{L}(\Phi(x_j), y_j),$$

where  $\mathcal{L}(\Phi(x_j), y_j)$  is a chosen loss function, such as mean squared error for regression or cross-entropy loss for classification.

To solve this optimization problem, gradient-based methods such as stochastic gradient descent (SGD) are commonly used. These methods require computing the gradients of the loss with respect to all network parameters:

$$\nabla_{\theta} \mathcal{L} = \left( \frac{\partial \mathcal{L}}{\partial W^{(1)}}, \frac{\partial \mathcal{L}}{\partial b^{(1)}}, \dots, \frac{\partial \mathcal{L}}{\partial W^{(L)}}, \frac{\partial \mathcal{L}}{\partial b^{(L)}} \right).$$

These gradients are efficiently computed using the backpropagation algorithm, which applies the chain rule recursively through the layers of the network. At each layer, we define the  $i$ -th layer output as  $A^{(i)} = W^{(i)}x^{(i-1)} + b^{(i)}$  and the  $i$ -th layer output after applying the activation function as  $x^{(i)} = \rho(A^{(i)})$ . Then the gradients with respect to parameters in layer  $i$  are:

$$\frac{\partial \mathcal{L}}{\partial W^{(i)}} = \frac{\partial \mathcal{L}}{\partial A^{(i)}} \cdot \frac{\partial A^{(i)}}{\partial W^{(i)}} = \frac{\partial \mathcal{L}}{\partial A^{(i)}} (x^{(i-1)})^\top, \quad \frac{\partial \mathcal{L}}{\partial b^{(i)}} = \frac{\partial \mathcal{L}}{\partial A^{(i)}} \cdot \frac{\partial A^{(i)}}{\partial b^{(i)}} = \frac{\partial \mathcal{L}}{\partial A^{(i)}}.$$

To compute  $\frac{\partial \mathcal{L}}{\partial A^{(i)}}$ , we use recursion from the next layer:

$$\frac{\partial \mathcal{L}}{\partial A^{(i)}} = \frac{\partial \mathcal{L}}{\partial x^{(i)}} \cdot \frac{\partial x^{(i)}}{\partial A^{(i)}} = \left( W^{(i+1)} \right)^\top \frac{\partial \mathcal{L}}{\partial A^{(i+1)}} \odot \rho'(A^{(i)}),$$

where  $\odot$  denotes elementwise multiplication.

Finally, we update the parameters using gradient descent:

$$W^{(i)} \leftarrow W^{(i)} - \eta \frac{\partial \mathcal{L}}{\partial W^{(i)}}, \quad b^{(i)} \leftarrow b^{(i)} - \eta \frac{\partial \mathcal{L}}{\partial b^{(i)}},$$

where  $\eta$  is the learning rate.

By combining their ability to approximate complex functions with efficient parameterization, neural networks serve as the backbone of a wide range of modern machine learning applications, including supervised learning, unsupervised learning, and generative modeling. In particular, they play a central role in diffusion-based generative models, where they are used to parameterize the reverse denoising process.

### 3 Related Work

Generative modeling is a central task in machine learning, with the goal of learning probability distributions from data and efficiently generating new samples. Classical generative models include Variational Autoencoders (VAEs) [4], Generative Adversarial Networks (GANs) [2], and normalizing flows [6]. Each of these approaches builds upon different modeling assumptions: VAEs introduce latent variables and optimize a variational lower bound; GANs frame generation as an adversarial game between a generator and discriminator; and flow-based models learn invertible mappings with tractable Jacobians. Despite their successes, these models often suffer from practical challenges such as training instability in GANs or restricted expressiveness in normalizing flows.

More recently, diffusion probabilistic models [3, 7] have emerged as a promising class of generative models. These models define a Markov chain that gradually diffuses data by adding Gaussian noise over  $T$  timesteps, and learn a reverse process to recover the original data distribution. The forward process is typically defined as:

$$q(x_t | x_{t-1}) = \mathcal{N}(x_t; \sqrt{1 - \beta_t} x_{t-1}, \beta_t \mathbf{I}),$$

where  $\beta_t$  denotes the variance schedule that controls the noise injected at each step. The reverse process is parameterized as a Gaussian distribution with learnable mean  $\mu_\theta(x_t, t)$ . Ho et al. [3] introduced a simplified training strategy, where a neural network is trained to directly predict the noise  $\epsilon_\theta(x_t, t)$  added during the forward process, allowing for a simplified loss function that ignores the original variational weighting terms.

Building on this foundation, several extensions have been proposed to improve the flexibility and sample quality of diffusion models. Nichol and Dhariwal [5] introduced learnable variance schedules and hybrid training losses. Other works have explored continuous-time diffusion models using score-based dynamics [8], improved samplers, and guidance techniques.

A key assumption in many standard diffusion models is that the reverse noise distribution has a fixed or diagonal covariance. Recent work has explored the implications of this assumption. For example, Everaert et al. [1] studied the effects of training diffusion models with isotropic Gaussian noise on data with anisotropic structure, showing that this mismatch can degrade generative performance. They proposed methods to better align the noise distribution with the data covariance using a learned linear transformation. While their work focused on high-dimensional image data, our approach investigates the role of fully learnable, input- and time-dependent covariance matrices in low-dimensional structured datasets. This allows for a detailed analysis of how the learned uncertainty behaves and aligns with the geometric features of the data.

### 4 Methods for Diffusion Models

Diffusion models synthesize data by gradually transforming pure noise into structured samples through a two-stage stochastic process: a forward diffusion process that progressively adds noise to the data, and a reverse denoising process that removes noise step by step. The core learning objective is to approximate this reverse process using a parameterized model trained on observed data. Once trained, the model can be used to generate realistic samples by simulating the learned reverse trajectory. Below, we provide an overview of this framework before detailing each component.

The overall procedure of diffusion models includes the following steps:

1. **Forward Diffusion Process:** Starting from original data  $\mathbf{x}_0$ , Gaussian noise is sequentially added over  $T$  timesteps, producing increasingly noisy data  $\mathbf{x}_1, \dots, \mathbf{x}_T$ .
2. **Reverse Denoising Process:** The model learns to reverse this noising process by estimating a sequence of denoising steps that map  $\mathbf{x}_T$  progressively back to  $\mathbf{x}_0$ .
3. **Training Objective:** The model is trained to minimize a loss function that measures the discrepancy between the learned reverse data distributions and true distribution derived from the forward diffusion process.
4. **Sampling Procedure:** After training, samples are generated by initializing with Gaussian noise and iteratively applying the learned reverse transitions to recover to recover  $x_0$  drawn from the distribution of the training samples.

Throughout this section, we use  $\mathcal{N}(\mathbf{x}; \mu, \Sigma)$  to denote a multivariate Gaussian distribution over  $\mathbf{x} \in \mathbb{R}^d$ , with mean  $\mu \in \mathbb{R}^d$  and covariance matrix  $\Sigma \in \mathbb{R}^{d \times d}$ :

$$\mathcal{N}(\mathbf{x}; \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu)^\top \Sigma^{-1}(\mathbf{x} - \mu)\right).$$

#### 4.1 Forward Diffusion Process

The forward process is modeled as a Markov chain in which data samples are added a small amount of Gaussian noise in successive steps. This process gradually transforms a data point  $\mathbf{x}_0$  into a noise sample  $\mathbf{x}_T$ .

Then the transition from timestep  $t - 1$  to  $t$  can be given by:

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

where:

- $\mathbf{x}_0$  denotes the original data,
- $\mathbf{x}_t$  denotes the noised data at timestep  $t$ ,
- $\beta_t$  denotes the variance schedule controlling the step sizes, typically increasing over time.

As  $T \rightarrow \infty$ , the diffused data  $\mathbf{x}_T$  becomes indistinguishable from pure Gaussian noise. The forward process defines each  $q(\mathbf{x}_t | \mathbf{x}_0)$  in closed form as a Gaussian as shown below, allowing the model to sample noisy data at arbitrary timesteps directly from  $\mathbf{x}_0$  without simulating the entire trajectory. This makes training efficient and tractable: the model can predict  $p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)$  and compare it against the true posterior  $q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)$ . This bypasses the need to marginalize over all paths and ensures the model learns to denoise inputs with varying noise levels.

A notable property of the forward process is that we can directly sample  $\mathbf{x}_t$  at any arbitrary time step  $t$  in a closed-form expression using the reparameterization trick. Let  $\alpha_t = 1 - \beta_t$  and  $\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$ . The forward process can then be expressed as:

$$\begin{aligned} \mathbf{x}_t &= \sqrt{\alpha_t} \mathbf{x}_{t-1} + \sqrt{1 - \alpha_t} \boldsymbol{\epsilon}_{t-1}, \quad \text{where } \boldsymbol{\epsilon}_{t-1}, \boldsymbol{\epsilon}_{t-2} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \\ &= \sqrt{\alpha_t \alpha_{t-1}} \mathbf{x}_{t-2} + \sqrt{1 - \alpha_t \alpha_{t-1}} \bar{\boldsymbol{\epsilon}}_{t-2}, \quad \text{where } \bar{\boldsymbol{\epsilon}}_{t-2} \text{ merges two Gaussians,} \\ &= \dots \\ &= \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \end{aligned}$$

Using this reparameterization, the marginal distribution of  $\mathbf{x}_t$  given the original data  $\mathbf{x}_0$  can be expressed as:

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

This property is crucial for efficiently computing the forward process and directly sampling from the forward diffusion distribution without requiring to simulate each intermediate time step.

## 4.2 Reverse Denoising Process

The reverse denoising process lies at the heart of diffusion-based generative modeling. It inverts the forward diffusion by gradually removing noise from a Gaussian sample to recover structured data. Mathematically, this is modeled as a Markov chain in which a noisy sample  $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  is iteratively transformed into a sample  $\mathbf{x}_0$  drawn from the data distribution.

The true reverse conditional probability  $q(\mathbf{x}_{t-1} | \mathbf{x}_t)$  is obtained by marginalizing over  $\mathbf{x}_0$ :

$$q(\mathbf{x}_{t-1} | \mathbf{x}_t) = \int q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) q(\mathbf{x}_0 | \mathbf{x}_t) d\mathbf{x}_0$$

However, this integral is intractable because it requires evaluating the true posterior  $q(\mathbf{x}_0 | \mathbf{x}_t)$ , which is unknown. To overcome this, diffusion models instead learn a parameterized distribution  $p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)$  to approximate  $q(\mathbf{x}_{t-1} | \mathbf{x}_t)$  directly.

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)$$

$$p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$$

During training, the model has access to the original data  $\mathbf{x}_0$ , which allows for a tractable approximation to the reverse transition. Specifically, we treat the conditional distribution  $q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)$  as a surrogate target. Using Bayes' theorem and the Markov property of the forward process, we can express this conditional as:

$$q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) = \frac{q(\mathbf{x}_t | \mathbf{x}_{t-1}) q(\mathbf{x}_{t-1} | \mathbf{x}_0)}{q(\mathbf{x}_t | \mathbf{x}_0)}.$$

where each term on the right-hand side is Gaussian:

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

$$q(\mathbf{x}_{t-1} | \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \sqrt{\bar{\alpha}_{t-1}} \mathbf{x}_0, (1 - \bar{\alpha}_{t-1}) \mathbf{I}),$$

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

We can now write the unnormalized product of Gaussians:

$$q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) \propto \exp \left( -\frac{1}{2} \left[ \frac{(\mathbf{x}_t - \sqrt{\alpha_t} \mathbf{x}_{t-1})^2}{\beta_t} + \frac{(\mathbf{x}_{t-1} - \sqrt{\bar{\alpha}_{t-1}} \mathbf{x}_0)^2}{1 - \bar{\alpha}_{t-1}} - \frac{(\mathbf{x}_t - \sqrt{\bar{\alpha}_t} \mathbf{x}_0)^2}{1 - \bar{\alpha}_t} \right] \right)$$

$$= \exp \left( -\frac{1}{2} \left[ \left( \frac{\alpha_t}{\beta_t} + \frac{1}{1 - \bar{\alpha}_{t-1}} \right) \mathbf{x}_{t-1}^2 - \left( \frac{2\sqrt{\alpha_t}}{\beta_t} \mathbf{x}_t + \frac{2\sqrt{\bar{\alpha}_{t-1}}}{1 - \bar{\alpha}_{t-1}} \mathbf{x}_0 \right) \mathbf{x}_{t-1} + C(\mathbf{x}_t, \mathbf{x}_0) \right] \right)$$

where  $C(\mathbf{x}_t, \mathbf{x}_0)$  is a constant that does not depend on  $\mathbf{x}_{t-1}$ .

By substituting the Gaussian forms of the forward conditional probabilities and simplifying, the mean and variance of the reverse process can be obtained as shown below.

$$\tilde{\beta}_t = \left( \frac{\alpha_t}{\beta_t} + \frac{1}{1 - \bar{\alpha}_{t-1}} \right)^{-1} = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \cdot \beta_t$$

$$\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = \tilde{\beta}_t \left( \frac{\sqrt{\alpha_t}}{\beta_t} \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1 - \bar{\alpha}_{t-1}} \mathbf{x}_0 \right)$$

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

$$= \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon}_t \right), \text{ substitute } \mathbf{x}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}} (\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}_t)$$

These expressions define the true reverse transition distribution conditioned on both the original and noisy data. Since  $\mathbf{x}_0$  is only available during training, this closed-form expression provides an analytically tractable learning target.

To approximate this process at test time when  $\mathbf{x}_0$  is unavailable, the model learns a neural network to parameterize the reverse distribution:

$$p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)),$$

where:

- $\mu_\theta(\mathbf{x}_t, t)$  is the predicted mean at timestep  $t$ ,
- $\Sigma_\theta(\mathbf{x}_t, t)$  is the learned covariance matrix modeling uncertainty.

### 4.3 Training Objective

#### 4.3.1 DDPM Loss

To train the reverse denoising process, the neural network is optimized to approximate the reverse conditional probability distributions  $p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)$ , which are parameterized as:

$$p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)).$$

The training objective is designed to ensure that the predicted mean  $\mu_\theta(\mathbf{x}_t, t)$  and covariance  $\Sigma_\theta(\mathbf{x}_t, t)$  align with the true reverse distribution.

During training, the Gaussian noise term can be reparameterized to simplify the loss calculation. Let  $\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0)$  denote the true mean of the reverse process:

$$\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon}_t \right),$$

where  $\boldsymbol{\epsilon}_t$  represents the Gaussian noise term in the forward process. The model then predicts the original data mean  $\mu_\theta(\mathbf{x}_t, t)$ , with the goal of approximating  $\tilde{\mu}_t$  as closely as possible. The reverse distribution can be written as:

$$\mathbf{x}_{t-1} \sim \mathcal{N} \left( \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon}_t \right), \Sigma_\theta(\mathbf{x}_t, t) \right).$$

The training loss  $L_t$  is parameterized to minimize the difference between the predicted mean  $\mu_\theta(\mathbf{x}_t, t)$  and the target mean  $\tilde{\mu}_t$ , weighted by the inverse covariance matrix  $\Sigma_\theta^{-1}$ :

$$L_t = \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}, t} \left[ \frac{1}{2 \|\Sigma_\theta(\mathbf{x}_t, t)\|_2} \|\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)\|_2^2 \right].$$

The reparameterization trick allows the loss to be rewritten directly in terms of the predicted noise  $\boldsymbol{\epsilon}_\theta$  and the true noise  $\boldsymbol{\epsilon}$ :

$$\begin{aligned} L_t &= \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}, t} \left[ \frac{(1 - \alpha_t)^2}{2\alpha_t(1 - \bar{\alpha}_t) \|\Sigma_\theta\|_2} \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\|_2^2 \right]. \\ &= \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}, t} \left[ \frac{(1 - \alpha_t)^2}{2\alpha_t(1 - \bar{\alpha}_t) \|\Sigma_\theta\|_2} \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}_t, t)\|_2^2 \right]. \end{aligned}$$

Empirical results [3] have shown that training diffusion models works effectively with a simplified objective that ignores the weighting term. This results in the commonly used noise-prediction loss:

$$\begin{aligned} L_t^{\text{simple}} &= \mathbb{E}_{t \sim [1, T], \mathbf{x}_0, \boldsymbol{\epsilon}, t} \left[ \|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\|_2^2 \right]. \\ &= \mathbb{E}_{t \sim [1, T], \mathbf{x}_0, \boldsymbol{\epsilon}, t} \left[ \|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}_t, t)\|_2^2 \right]. \end{aligned}$$

This simplified loss function trains the neural network to predict the noise  $\boldsymbol{\epsilon}$  added during the forward process. By learning to denoise the noisy sample  $\mathbf{x}_t$ , the network indirectly learns to approximate the true reverse conditional distribution, enabling an effective reconstruction of the original data  $\mathbf{x}_0$ .

---

**Algorithm 1** Training with Scalar Variance

---

- 1: **repeat**
  - 2:    $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
  - 3:    $t \sim \text{Uniform}(\{1, \dots, T\})$
  - 4:    $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}$
  - 5:   Take gradient descent step on:  
 $\nabla_{\theta} \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\theta}(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}, t)\|^2$
  - 6: **until** converged
- 

### 4.3.2 Our Loss with Learned Covariance

In our method, we learn both the mean  $\mu_{\theta}(\mathbf{x}_t, t)$  and a full, time- and input-dependent covariance matrix  $\Sigma_{\theta}(\mathbf{x}_t, t)$  to parameterize the reverse transition:

$$p_{\theta}(\mathbf{x}_{t-1} | \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_{\theta}(\mathbf{x}_t, t), \Sigma_{\theta}(\mathbf{x}_t, t)).$$

During training, we have access to the true reverse posterior

$$q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_t, \Sigma_t),$$

where  $\mu_t$  and  $\Sigma_t$  are the empirical mean and covariance computed at timestep  $t$ . This allows us to match the learned distribution to the target by minimizing the KL divergence between Gaussians:

$$\mathcal{L}_{\text{KL}} = \mathbb{E}_{\mathbf{x}_0, t} [D_{\text{KL}}(\mathcal{N}(\mu_{\theta}, \Sigma_{\theta}) \| \mathcal{N}(\mu_t, \Sigma_t))].$$

To preserve compatibility with the original DDPM noise-prediction training, we retain the standard denoising objective:

$$\mathcal{L}_t^{\text{simple}} = \mathbb{E}_{t \sim [1, T], \mathbf{x}_0, \boldsymbol{\epsilon}, t} [\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_{\theta}(\mathbf{x}_t, t)\|^2].$$

Combining both objectives, our full training loss becomes:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_t^{\text{simple}} + \lambda \cdot \mathcal{L}_{\text{KL}},$$

where  $\lambda > 0$  is a regularization weight balancing denoising fidelity and uncertainty matching.

The training procedure is summarized below.

---

**Algorithm 2** Training with Learned Covariance

---

- 1: **repeat**
  - 2:    $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
  - 3:    $t \sim \text{Uniform}(\{1, \dots, T\})$
  - 4:    $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}$
  - 5:   Take gradient descent step on:  
 $\nabla_{\theta} \left( \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\theta}(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}, t)\|^2 + \lambda \cdot D_{\text{KL}}(\mathcal{N}(\mu_{\theta}, \Sigma_{\theta}) \| \mathcal{N}(\mu_t, \Sigma_t)) \right)$
  - 6: **until** converged
- 

## 4.4 Sampling Procedure

### 4.4.1 Sampling in DDPM

Once the reverse process  $p_{\theta}(\mathbf{x}_{t-1} | \mathbf{x}_t)$  has been learned, we can generate new data by sampling from the model. The generative sampling procedure starts with a Gaussian noise  $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  and iteratively denoises it using the learned reverse distributions:

$$\mathbf{x}_{t-1} \sim \mathcal{N}(\mu_{\theta}(\mathbf{x}_t, t), \Sigma_{\theta}(\mathbf{x}_t, t)), \quad \text{for } t = T, T-1, \dots, 1.$$

In the original DDPM framework, the variance is modeled as a learned scalar. The most common setup chooses a fixed scalar schedule:

$$\Sigma_{\theta}(\mathbf{x}_t, t) = \sigma_t^2 \mathbf{I},$$

which leads to the following sampling procedure:

$$\mathbf{x}_{t-1} = \mu_\theta(\mathbf{x}_t, t) + \sigma_t \cdot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).$$

---

**Algorithm 3** Sampling with Scalar Variance
 

---

```

1:  $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
2: for  $t = T, \dots, 1$  do
3:    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  if  $t > 1$ , else  $\mathbf{z} = \mathbf{0}$ 
4:    $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\alpha_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$ 
5: end for
6: return  $\mathbf{x}_0$ 

```

---

#### 4.4.2 Sampling with Learned Covariance

In our proposed method, the reverse variance is not restricted to a fixed scalar. Instead, we allow the model to predict a full, time- and input-dependent covariance matrix:

$$\Sigma_\theta(\mathbf{x}_t, t) \in \mathbb{R}^{d \times d}, \quad \Sigma_\theta(\mathbf{x}_t, t) = \Sigma_\theta(\mathbf{x}_t, t)^\top, \quad \Sigma_\theta(\mathbf{x}_t, t) \succ 0.$$

This enables the model to represent anisotropic and correlated noise patterns, adapting the sampling direction and scale to the geometry of the data at each timestep. Sampling is performed using:

$$\mathbf{x}_{t-1} = \mu_\theta(\mathbf{x}_t, t) + \Sigma_\theta^{1/2}(\mathbf{x}_t, t) \cdot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}),$$

where  $\Sigma_\theta^{1/2}$  is a matrix square root (e.g., Cholesky factor) of the predicted covariance.

This iterative process from  $t = T$  down to  $t = 0$  gradually transforms pure Gaussian noise into a structured sample  $\mathbf{x}_0$  drawn from the learned data distribution.

---

**Algorithm 4** Sampling with Learned Covariance
 

---

```

1: Initialize  $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
2: for  $t = T, \dots, 1$  do
3:    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  if  $t > 1$ , else  $\mathbf{z} = \mathbf{0}$ 
4:   Compute mean  $\mu_\theta(\mathbf{x}_t, t)$  and covariance  $\Sigma_\theta(\mathbf{x}_t, t)$ 
5:   Sample:  $\mathbf{x}_{t-1} = \mu_\theta(\mathbf{x}_t, t) + \Sigma_\theta^{1/2}(\mathbf{x}_t, t) \cdot \mathbf{z}$ 
6: end for
7: return  $\mathbf{x}_0$ 

```

---

## 5 Experiments

### 5.1 Datasets

To investigate the effect of learning a full covariance matrix in diffusion models, we conduct experiments on several synthetic two-dimensional datasets. In total, we consider four datasets: **Swiss roll**, **S-curve**, **two moons**, and **concentric circles**. These datasets capture diverse nonlinear geometric structures—such as branching, curvature, and symmetry—while remaining low-dimensional and visually interpretable, making them ideal for analyzing the impact of noise modeling in diffusion processes.

Not all datasets are used in every experiment. The Swiss roll and S-curve datasets serve as the primary test cases for most evaluations due to their complex structure. The Moons and Circles datasets are included specifically in the multi-label reconstruction experiment to test generalization across different geometric distributions.

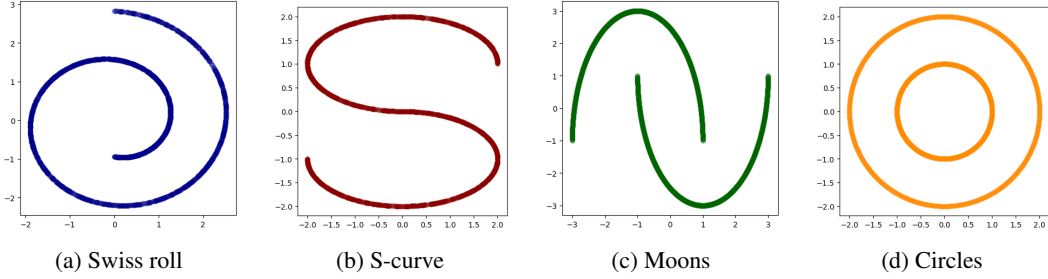


Figure 1: Four synthetic 2D datasets used in the experiments.

## 5.2 Experimental Setup

We follow the DDPM framework, where the forward process diffuses the original data over  $T = 500$  and  $T = 700$  timesteps using a fixed variance schedule. Specifically, we use a sigmoid schedule for the noise scale, with  $\beta_t$  ranging from  $1 \times 10^{-5}$  to 0.03. The derived quantities  $\alpha_t = 1 - \beta_t$  and  $\bar{\alpha}_t = \prod_{s=1}^t \alpha_s$  define the diffusion and denoising dynamics used throughout.

Two neural networks are used in our model:

- The noise prediction network  $\epsilon_\theta(\mathbf{x}_t, t)$ , which estimates the noise added at each timestep.
- The covariance prediction network  $\Sigma_\theta(\mathbf{x}_t, t)$ , which outputs a full  $2 \times 2$  positive-definite matrix modeled via a parameterized Cholesky factor.

Both networks are implemented as fully connected multilayer perceptrons (MLPs) with four hidden layers of width 128 and ReLU activations. Timestep information is embedded as a scalar and concatenated to the input.

We train using the Adam optimizer with a learning rate of  $10^{-3}$ , batch size of 256, and for 2,000 epochs. The loss for the baseline DDPM model is the standard mean squared error on noise prediction. For our method, we add a KL divergence loss between the predicted covariance matrix and the empirical covariance  $\hat{\Sigma}_t$ , computed from batch noise residuals.

We conduct a series of experiments to systematically evaluate our model and analyze its behavior under different settings:

- **Baseline comparison:** Comparison between the standard DDPM with fixed diagonal covariance and our model that learns a full input- and time-dependent covariance matrix.
- **Labeled vs. unlabeled training:** Evaluation of how access to class label information during training affects reconstruction quality, by comparing models trained with known class labels (labeled training) to models trained without label information (unlabeled training).
- **Multi-label generation:** Assessment of the model’s conditional generation ability by assigning multiple class labels to different datasets and evaluating its capacity to learn multiple shape classes.
- **Sampling noise scaling:** Investigation of the effect of modifying the noise magnitude during sampling. A scaling factor  $\lambda$  is applied to the noise term  $\epsilon$ , with values  $\lambda \in \{0, 0.2, 0.4, \dots, 2\}$ , allowing us to study the trade-off between sample diversity and generation stability.

## 5.3 Baseline Comparison

### 5.3.1 Baseline Results: DDPM with Scalar Variance

As a baseline, we implement the standard Denoising Diffusion Probabilistic Model (DDPM) with a fixed, scalar noise schedule. The reverse process is modeled as:

$$p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \beta_t \mathbf{I}),$$

where  $\mu_\theta(\mathbf{x}_t, t)$  is computed using the predicted noise  $\epsilon_\theta(\mathbf{x}_t, t)$ :

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

We train the noise-prediction model using the standard DDPM loss:

$$\mathcal{L}_{\text{DDPM}} = \mathbb{E}_{\mathbf{x}_0, t, \epsilon} \left[ \|\epsilon - \epsilon_\theta(\mathbf{x}_t, t)\|^2 \right],$$

where  $\mathbf{x}_t = \sqrt{\alpha_t} \mathbf{x}_0 + \sqrt{1 - \alpha_t} \epsilon$ , and  $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ .

We visualize the sampling trajectory of the baseline DDPM model as it denoises from pure Gaussian noise  $\mathbf{x}_T \sim \mathcal{N}(0, I)$  back to the original data distribution. As shown in Figure 2, the model progressively refines the noisy sample toward reconstructed outputs over 700 timesteps, where each subfigure corresponds to an intermediate timestep.

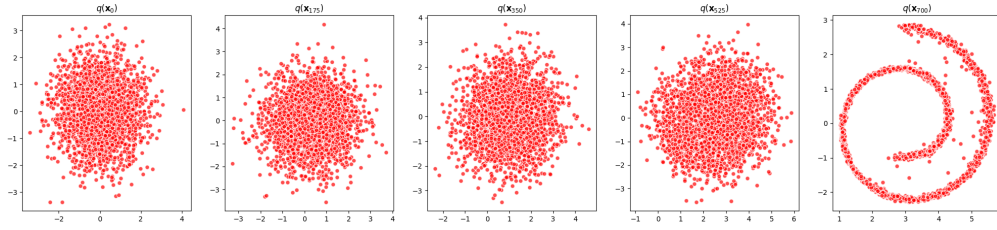


Figure 2: Sampling trajectory from noise to reconstructed data using the baseline DDPM model.

These results establish a baseline performance and motivate the need for more expressive noise modeling in the reverse process, especially in data with anisotropic uncertainty or geometry-dependent structure.

### 5.3.2 Our Method: Learning Full Covariance

In contrast to the baseline DDPM, which assumes a fixed isotropic noise distribution in the reverse process, we consider to learn a full, input-dependent covariance matrix at each timestep. Specifically, our model parameterizes the reverse distribution as:

$$p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)),$$

where the mean  $\mu_\theta(\mathbf{x}_t, t)$  is derived from the predicted noise  $\epsilon_\theta(\mathbf{x}_t, t)$  via the standard DDPM formula, and the covariance  $\Sigma_\theta(\mathbf{x}_t, t)$  is predicted using a separate neural network that outputs a parameterized Cholesky decomposition to ensure positive definiteness.

To train this model, we retain the noise prediction loss used in DDPM:

$$\mathcal{L}_t^{\text{simple}} = \mathbb{E}_{\mathbf{x}_0, t, \epsilon} \left[ \|\epsilon - \epsilon_\theta(\mathbf{x}_t, t)\|^2 \right],$$

but with an additional regularization term that encourages the predicted covariance matrix to match the empirical covariance of the true noise.

Let  $\epsilon_t = (\mathbf{x}_t - \sqrt{\alpha_t} \mathbf{x}_0) / \sqrt{1 - \alpha_t}$  denote the actual noise added during the forward process. At each timestep, we compute the empirical sample covariance of  $\epsilon_t$  over the batch, and define the KL divergence:

$$\mathcal{L}_{\text{KL}} = \mathbb{E}_{\mathbf{x}_t, t} [D_{\text{KL}}(\mathcal{N}(0, \Sigma_\theta(\mathbf{x}_t, t)) \| \mathcal{N}(0, \Sigma_t))],$$

where  $\Sigma_t$  is the empirical batch covariance of  $\epsilon_t$  at timestep  $t$ . This loss encourages the model to learn uncertainty that reflects the local structure of the data at different levels of corruption.

The full training objective becomes:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_t^{\text{simple}} + \lambda \cdot \mathcal{L}_{\text{KL}},$$

where  $\lambda > 0$  is a weighting hyperparameter.

We also visualize the sampling trajectory of our model with learned covariance as it iteratively denoises from pure Gaussian noise  $\mathbf{x}_T \sim \mathcal{N}(0, I)$  back to the original data distribution (see Figure 3). Throughout the reverse process, the model refines noisy samples while adaptively modulating uncertainty through the learned covariance at each step.

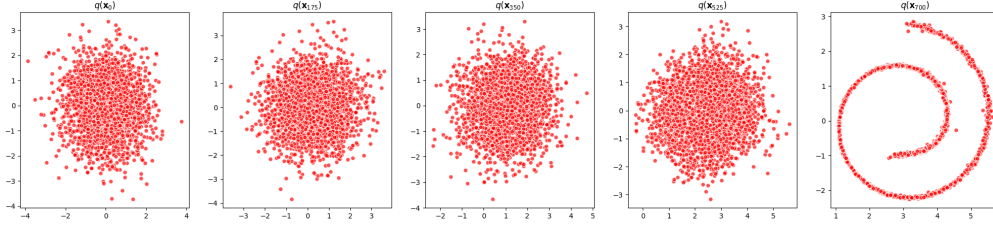


Figure 3: Sampling trajectory from noise to reconstructed data using our covariance-based model.

In addition to the full trajectory, we below provide a more direct visual comparison of the final generated samples between the baseline DDPM and our model, shown in Figure 4. The learned covariance produces sharper boundaries, improved adaptation, and more faithful reconstruction of complex geometric structures.

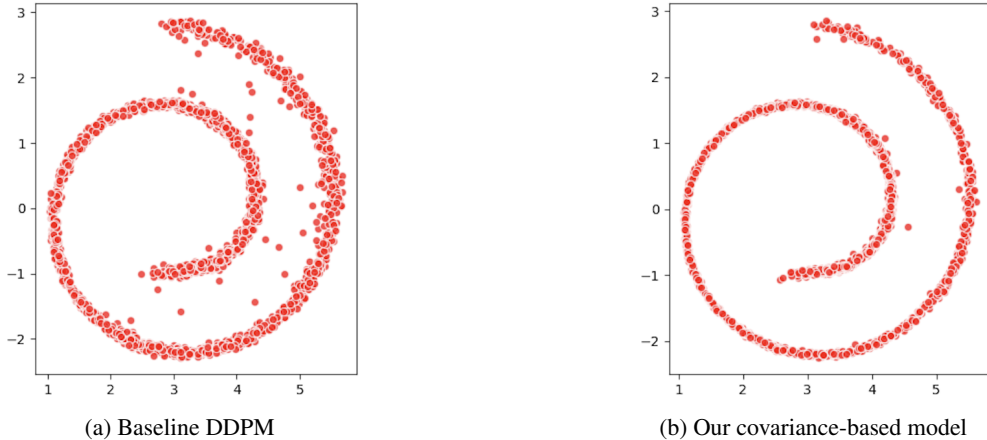


Figure 4: Comparison of final reconstructed samples between the baseline DDPM and our model.

#### 5.4 Labeled vs. Unlabeled Training

In this thesis, the term “labeled” refers specifically to whether class information (i.e., geometric group identity) is provided during training. Both labeled and unlabeled settings have full access to the original training data  $\mathbf{x}_0$ , but differ in whether additional class labels  $y$  are available:

- **Unlabeled training:** The model observes only  $\{\mathbf{x}_t\}$  without any class identity. It learns a single unconditional generative model that captures the entire mixture of geometric structures. The training loss is:

$$\mathcal{L}_{\text{simple}} = \mathbb{E}_{\mathbf{x}_0, t, \epsilon} \left[ \|\epsilon - \epsilon_\theta(\mathbf{x}_t, t)\|^2 \right].$$

- **Labeled training:** Each data point  $\mathbf{x}_0$  is paired with its class label  $y$ , which indicates the geometric group (e.g., Swiss Roll, S-curve, Moons, Circles). The model observes pairs  $\{x_t, y\}$ , where the label  $y$  is provided as an additional input to the model at both training and sampling time. The model thus learns conditional generation given class identity. The loss becomes:

$$\mathcal{L}_{\text{simple}}^{\text{labeled}} = \mathbb{E}_{\mathbf{x}_0, t, \epsilon, y} \left[ \|\epsilon - \epsilon_\theta(\mathbf{x}_t, t, y)\|^2 \right].$$

This formulation allows us to directly evaluate the effect of label conditioning on the model’s ability to disentangle and reconstruct distinct geometric structures. Figure 5 presents reconstructions under both training regimes. The labeled model (bottom) successfully captures clear geometric boundaries and accurately reproduces class-specific patterns that closely match the original data distribution. In contrast, while the unlabeled model (top) still produces globally coherent samples, its outputs occasionally exhibit ambiguity, with partial blending or distortions between shapes.

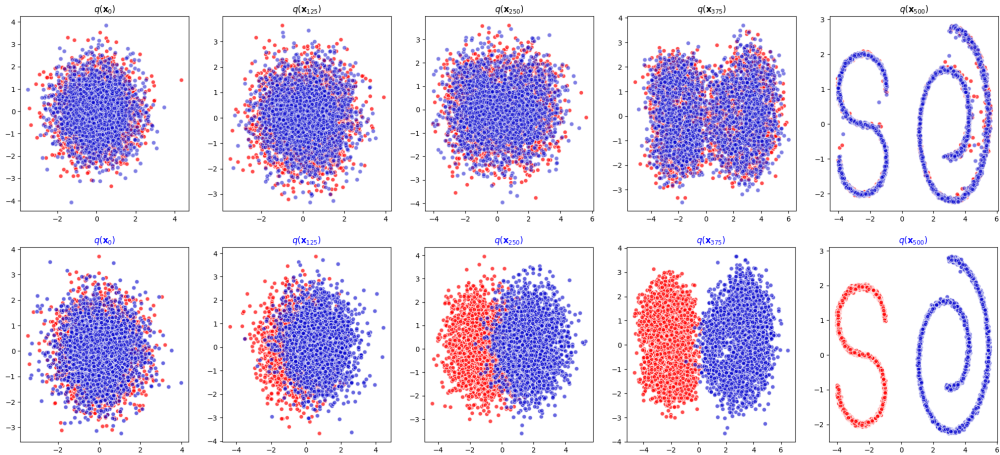


Figure 5: Reconstructions from models trained without (top) and with (bottom) class labels.

Overall, both models demonstrate the inherent robustness of diffusion models in generating plausible samples across diverse shapes. However, the inclusion of class labels allows the model to better preserve class-specific structures, maintain clearer separation between different geometric patterns, and prevent unintended blending between distinct groups. This highlights the advantage of conditional supervision in structured data generation, enabling finer control over the output distribution and improving reconstruction quality for diverse, well-separated datasets.

### 5.5 Multi-Label Generation with Overlapping Shape Categories

To further evaluate the model’s ability to generate structured data under more challenging scenarios, we conduct experiments in a multi-label setting where more geometric shapes are combined into a single dataset. Specifically, we consider four shape categories: **Swiss roll**, **S-curve**, **Moons**, and **Circles**.

In previous experiments, each dataset was arranged such that each class occupied distinct, non-overlapping regions in 2D space. In this setting, however, samples from all classes are mixed together and distributed in overlapping regions, introducing additional complexity for both disentanglement and reconstruction.

During training, each data point is assigned a class label  $y \in \{\text{Swiss roll, S-curve, Moons, Circles}\}$ , which serves as the conditioning input to the model. This allows the model to generate class-specific reconstructions even when samples from different classes are interleaved in the same spatial domain.

Figure 6 presents the conditional reconstructions for each shape category. Despite the substantial geometric overlap between classes, the model successfully disentangles and reconstructs each structure when provided with the corresponding label. The generated samples preserve curvature, density, and topological characteristics across all classes.

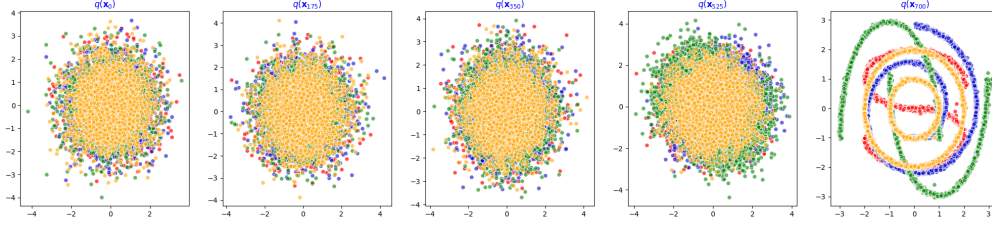


Figure 6: Conditional reconstructions with overlapping shape categories (Swiss roll, S-curve, Moons, and Circles).

These results demonstrate that, even under greater diversity and significant geometric overlap, conditioning on class labels allows the model to effectively disentangle distinct data modes and preserve class-specific structures in its generations. Moreover, they further highlight the generality of our covariance-based diffusion framework, which remains robust in capturing complex geometric patterns as the number and diversity of classes increase.

### 5.6 Effect of Noise Scaling in the Sampling Process

In the standard DDPM sampling procedure, the sample  $\mathbf{x}_{t-1}$  is drawn from a Gaussian distribution centered at the predicted mean  $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)$ , with variance  $\sigma_t^2$  determined by the noise schedule. The formulation is given by:

$$\mathbf{x}_{t-1} = \boldsymbol{\mu}_\theta(\mathbf{x}_t, t) + \sigma_t \cdot \mathbf{z}, \quad \mathbf{z} \sim \mathcal{N}(0, \mathbf{I}).$$

To investigate how the amount of stochasticity on noise affects sample quality and structure, we introduce a scaling factor  $\lambda$  to control the injected noise during sampling:

$$\mathbf{x}_{t-1} = \boldsymbol{\mu}_\theta(\mathbf{x}_t, t) + \lambda \cdot \sigma_t \cdot \mathbf{z}.$$

This modification allows us to adjust the intensity of the injected noise, thereby controlling the trade-off between diversity and fidelity during generation.

We experiment with 11 values of  $\lambda$  ranging from fully deterministic ( $\lambda = 0$ ) to highly stochastic ( $\lambda = 2.0$ ):

$$\lambda \in \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0\},$$

which we categorize into four parts for qualitative comparison: (1) no noise ( $\lambda = 0$ ), (2) reduced noise ( $0 < \lambda < 1$ ), (3) standard noise ( $\lambda = 1$ ), and (4) amplified noise ( $\lambda > 1$ ).

**No noise ( $\lambda = 0$ )** With  $\lambda = 0$ , the sampling becomes deterministic: the model follows the learned mean trajectory without any randomness. While reconstructions tend to be sharp, they often do not cover the full structure or diversity of the data.

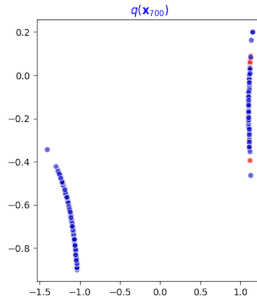


Figure 7: Sampling with  $\lambda = 0$ : fully deterministic trajectories.

**Reduced noise ( $0 < \lambda < 1$ )** Introducing a small amount of noise increases variability while maintaining coherent structure. We observe that noise levels in the range  $\lambda = 0.6\text{--}0.8$  yield a favorable balance between diversity and geometric fidelity.

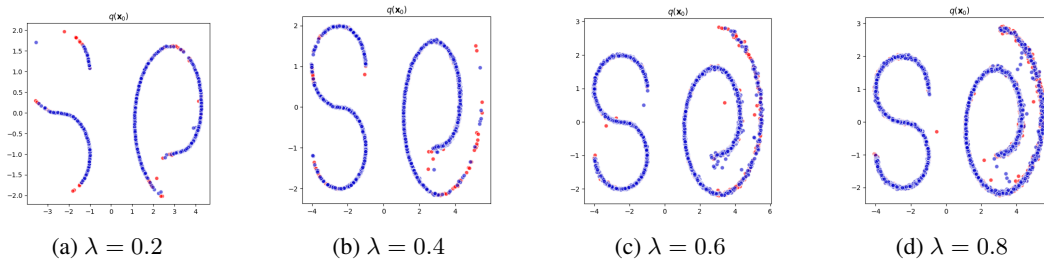


Figure 8: Sampling with reduced noise. Moderate stochasticity improves realism without significant distortion.

**Standard noise ( $\lambda = 1.0$ )** This corresponds to the default DDPM sampling behavior. It provides a balanced compromise between diversity and structure preservation and serves as the baseline for comparison.

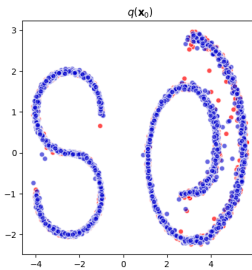


Figure 9: Sampling with  $\lambda = 1$ : standard DDPM noise level.

**Amplified noise ( $\lambda > 1$ )** Scaling the noise beyond its original value leads to increasingly random samples. While this can increase sample variability, it also tends to degrade structural integrity and produce visibly distorted outputs.

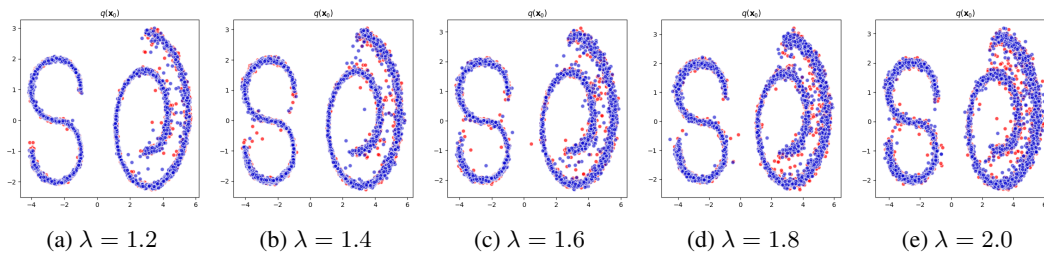


Figure 10: Sampling with amplified noise. Excessive randomness leads to degraded geometric coherence.

In summary, our experiments demonstrate that the noise scale  $\lambda$  during sampling plays a crucial role in controlling the trade-off between sample fidelity and diversity. Reducing  $\lambda$  sharpens the generated structures and improves reconstruction quality by suppressing stochastic variation. In contrast, increasing  $\lambda$  promotes sample diversity but may introduce geometric distortions or blur fine details. These results suggest that  $\lambda$  may serve as a valuable hyperparameter for tuning the balance between precision and variability in diffusion-based generation.

## 6 Conclusion

In this thesis, we consider a diffusion-based generative modeling framework that extends conventional DDPMs by learning full anisotropic, input- and time-dependent covariance matrices in the reverse process. By moving beyond the standard assumption of fixed diagonal or isotropic noise, our approach enables the model to capture richer uncertainty structures and more accurately reflect complex dependencies in the data distribution. We provide complete mathematical derivations for both forward and reverse processes and introduce a joint training objective that combines noise prediction loss with KL-divergence-based covariance prediction.

We validated the effectiveness of our method on multiple 2D synthetic datasets, demonstrating improved reconstruction fidelity, better preservation of geometric structure, and enhanced separation under label conditioning. Additionally, we analyzed the effects of varying noise scaling during sampling, revealing controllable trade-offs between sample diversity and structural sharpness.

This work lays a foundation for further exploration of covariance modeling in diffusion processes. Future research may extend this approach to higher-dimensional and real-world data, integrate it with more advanced neural network architectures, and investigate its potential in combination with emerging developments in generative modeling and diffusion-based frameworks.

## References

- [1] Simon Everaert, Julien Virmaux, Pierre Baque, and Pascal Fua. Covariance mismatch in diffusion models. *arXiv preprint arXiv:2305.14288*, 2023.
- [2] Ian Goodfellow et al. Generative adversarial nets. In *NeurIPS*, 2014.
- [3] Jonathan Ho et al. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.
- [4] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [5] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *ICML*, 2021.
- [6] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *ICML*, 2015.
- [7] Jascha Sohl-Dickstein et al. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2015.
- [8] Yang Song, Jascha Sohl-Dickstein, et al. Score-based generative modeling through stochastic differential equations. *ICLR*, 2021.