UNIVERSITY OF COPENHAGEN
FACULTY OF SCIENCE

Machine Learning

Embedded Expression Constraints in Upscaling Diffusion Models

Simon Lolk Pallesen

Supervised by Julius Bier Kirkegaard

May 2025

Simon Lolk Pallesen
Embedded Expression Constraints in Upscaling Diffusion Models
Machine Learning, May 2025
Supervisors: Julius Bier Kirkegaard

University of Copenhagen
Department of Computer Science
Master Degree in Data Science
Universitetsparken 1
2100 Copenhagen N

Preface

This Master's thesis is written by Simon Lolk Pallesen as part of the masters degree in Computer Science. The project was conducted from January 1th, 2025 to May the 31th, 2025 at the Department of Computer Science at the University of Copenhagen (DIKU). This thesis is 30 ECTS-points with the guidance of Julius Bier Kirkegaard as supervisor.

Abstract

TODO

Contents

Introduction

1

Denoising Diffusion Probabilistic Models (DDPMs), first introduced by Ho et al. [HJA20], have emerged as a leading class of generative models, setting new standards in fields like image synthesis, inpainting, and super-resolution. These models operate through a two-phase process: a forward diffusion step, which gradually adds Gaussian noise to data over a series of timesteps, and a learned reverse process, which denoises this noisy input to recover data samples.

This reverse denoising process is parameterized by a neural network that learns to predict the noise component added at each timestep. However, sampling from the model is deliberately not deterministic. Even when the model accurately predicts the noise (denoted by ϵ_θ), the generation process still involves injecting fresh Gaussian noise $\mathbf{z}$ at each step, scaled by a time-dependent variance term. This added noise is sampled from an isotropic Gaussian, meaning each component is drawn independently and without regard for any structure or dependencies in the underlying data.

While this design supports the flexibility and expressiveness of DDPMs, it introduces a limitation in settings where the data obeys known structure, physical laws, or consistency constraints. Because the noise is sampled independently across all dimensions, it may disrupt relationships the model has implicitly learned to preserve, particularly in domains that demand strict adherence to local or global constraints.

In this thesis, we will discuss and formalize a method for hijacking the diffusion sampling process to enforce known constraints during generation. Our approach modifies only the sampling step, without retraining the model or altering its architecture, to better align outputs with domain-specific structures.

In Section 3, we define the theoretical and mathematical framework behind DDPMs, and explain where and how our method integrates into the sampling process. In Section 4, we introduce relevant metrics and strategies for validating whether generated samples respect intended constraints. In Sections 5 and 6, we detail the experiments that apply our method to real-world scenarios. Finally, in Sections 7 and 8, we reflect on the implications of our findings, discuss limitations and future directions, and summarize our contributions.

Problem Statement

2

Diffusion models have shown remarkable success across a range of generative tasks, including image synthesis, denoising, and super-resolution. These models are prized for their ability to approximate complex data distributions through a learned sequence of denoising steps. However, they are fundamentally probabilistic in nature, particularly during sampling, where Gaussian noise is injected at each step of the reverse process to introduce variability and support a broad output distribution.

While this stochasticity is essential to model expressiveness, it also introduces a critical limitation: diffusion models do not inherently enforce structural or domain-specific constraints that may be known a priori. This becomes particularly problematic in tasks where physical laws or spatial relationships must be preserved.

For example, in image super-resolution, a low-resolution pixel can be interpreted as the average of a 2×2 patch in the corresponding high-resolution image. If a generated high-resolution image violates this averaging constraint, the result may look plausible to the model, but remain inconsistent with reality, under the actual low-resolution input. Similarly, in fluid simulation tasks, generated velocity or density fields must often conserve quantities such as mass or momentum. Without explicit enforcement of these physical laws, the generated outputs may appear realistic in isolation, but fail to respect essential conservation principles.

These violations are not due to flaws in the model's training or representational capacity, but rather stem from the unconstrained nature of the sampling process itself. The model may learn to approximate the true distribution accurately, yet still produce individual samples that contradict known constraints.

The central problem this thesis addresses is therefore:

> *How can we guide the generative process of diffusion models to respect known structural or physical constraints, without retraining the model or modifying its architecture?*

We hypothesize that constraint violations are not primarily caused by poor model predictions, but rather by the unstructured nature of the added variance noise during sampling. The network's prediction μ_θ can, in principle, learn to preserve constraints if they are reflected in the data distribution. However, the independently sampled noise vector $\mathbf{z}$ may counteract this, pushing the sample off the domain of constraint-satisfying solutions.

To address this, we propose a novel sampling modification: instead of sampling the entire variance noise vector from an isotropic Gaussian, we deterministically sample from a conditioned gaussian distribution given the model output μ_θ and a tractable condition expression, to ensure that the

truth value of the condition is handled uniformly, and not unfairly solved by a single pixel. For example, in the super-resolution setting, if we require that the average of four high-resolution pixels equals a known low-resolution value, we can conditionally sample three pixels as and solve for the fourth. This way, the stochastic nature of generation is preserved while ensuring constraint satisfaction.

Unlike prior work, our method is not limited to a particular application or implicit constraint. It provides a general framework for enforcing arbitrary linear constraints during the sampling process of diffusion models. The aim of this thesis is to formalize this constrained noise sampling method, demonstrate its general applicability, and empirically evaluate its effectiveness in diverse tasks.

Related work Recent research has recognized the potential of modifying the diffusion sampling process, without retraining or altering the model architecture, to improve sample quality or enforce specific output characteristics. For instance, Tailanián et al. [Tai+24] introduce *diffusion purification* for image counter-forensics, where the forward and reverse diffusion processes are manipulated to erase forensic traces from images. Their approach demonstrates that tampering with the sampling process alone can significantly alter the statistical properties of generated images, outperforming prior counter-forensic techniques in both effectiveness and visual fidelity.

Similarly, Hwang et al. [HPJ24] propose *upsample guidance*, a method that adapts pre-trained diffusion models to generate higher-resolution images by adding a single guidance term during sampling. This technique does not require any additional training or architectural modifications, and can be applied to a wide variety of diffusion models, including those operating in pixel, latent, or video domains. The success of upsample guidance underscores the power of targeted sampling modifications for enhancing diffusion model outputs.

While these works demonstrate the effectiveness of sampling-based interventions, they are typically tailored to specific applications, such as resolution upscaling. Their constraints are often implicit or limited to a particular domain. We propose a theoretical framework that is applicable in a general setting.

2.1 Problem Domains

To evaluate the effectiveness and generality of our proposed method, we apply it across three distinct problem domains, each posing a unique structural or physical constraint that aligns well with our framework. These domains span from visual super-resolution to physical simulations and stochastic processes. Our goal is not to compete on benchmark performance, but rather to demonstrate how the method adapts to different types of constraints without requiring changes to the underlying diffusion model architecture or retraining strategy.

The selected problems share a common feature: they each impose hard constraints that must be satisfied at every step of the generative process. These constraints are either local, as in the case of image super-resolution, or global, as in physical simulations and time-series modeling.

Figure 2.1 provides example outputs from the datasets used in each setting.

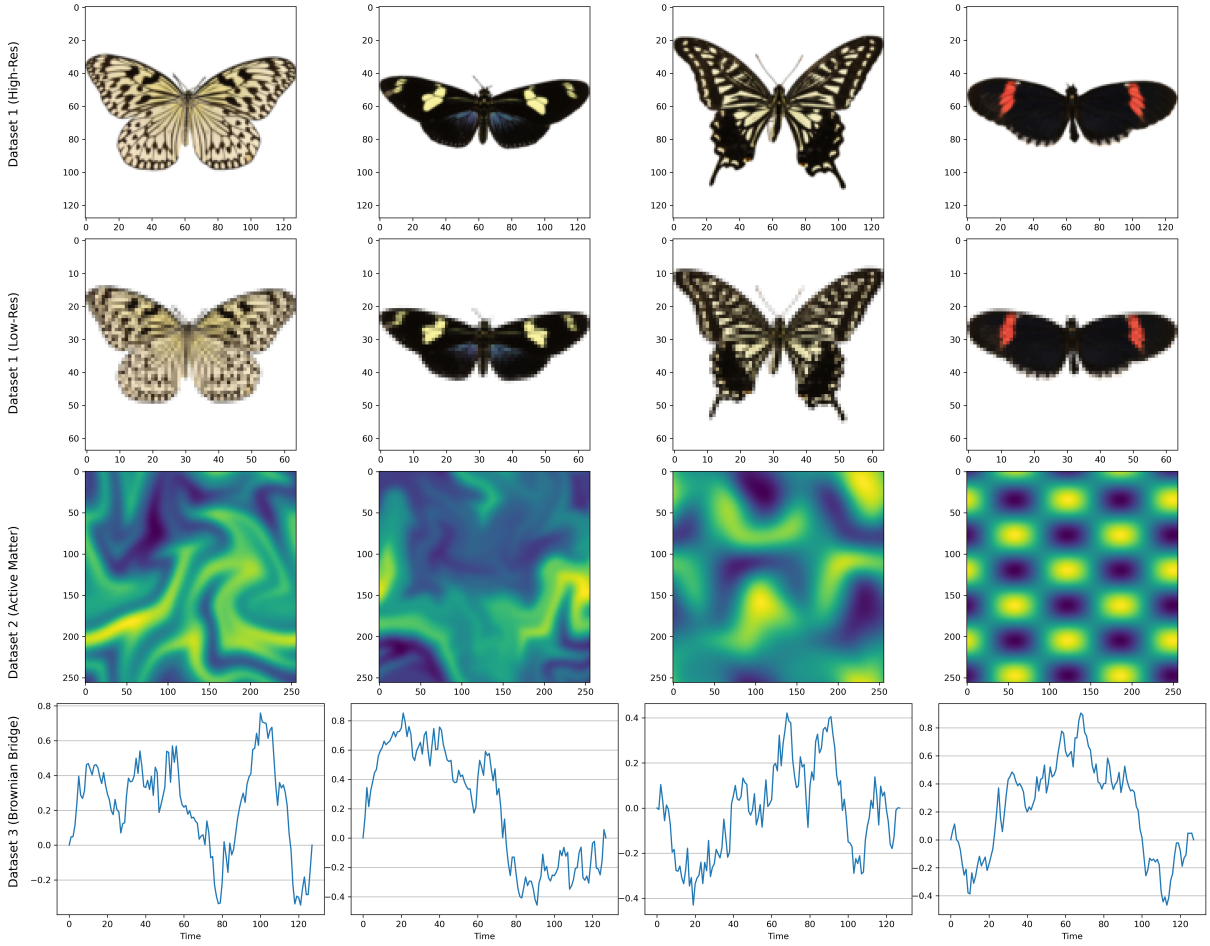

Figure 2.1: Sample images and trajectories from the datasets used in our experiments. From left to right: Super-resolution (butterflies), spatial concentration fields, and Brownian bridges.

The first domain involves super-resolution, where the task is to reconstruct a high-resolution image from a low-resolution input. A natural averaging constraint exists between corresponding pixel blocks, making it an ideal testbed for our method.

The second domain involves physical simulations governed by conservation laws, specifically concentration conservation in fluid-like systems. These simulations provide spatiotemporal fields where the average concentration must remain fixed, making global constraint enforcement critical.

Finally, we apply our method to a 1D Brownian bridge process. Which is a type of stochastic time-series that is constrained to return to a known endpoint.

These domains allow us to showcase the method's flexibility across image, scientific, and time-series applications, all while preserving hard constraints through principled, constraint-aware sampling.

Theoretical Framework

3

This chapter outlines the theoretical underpinnings essential for understanding and implementing a Denoising Diffusion Probabilistic Model (DDPM), and how we plan on hijacking the diffusion process in three separate scenarios. The discussion begins with an overview of the diffusion processes that form the core of the model, followed by an exploration of the training objectives that guide the learning procedure. Subsequently, sampling methods for data generation are introduced, and the chapter concludes with an examination of the model architecture.

3.1 Denoising Diffusion Probabilistic Model

Denoising Diffusion Probabilistic Models (DDPMs) are built upon two fundamental processes: the *forward diffusion process* and the *reverse diffusion process*. The forward diffusion process serves to progressively corrupt an input image by adding Gaussian noise in a stepwise fashion. Over a fixed number of steps T, the image transitions from its original distribution to a standard Gaussian distribution, converging to pure isotropic noise $\mathcal{N}(\mathbf{0}, \mathbf{I})$.

Conversely, the reverse diffusion process aims to reconstruct or generate data by denoising this noisy representation step by step. Starting from a sample drawn from $\mathcal{N}(\mathbf{0}, \mathbf{I})$, the process iteratively removes noise, thereby recovering a clean image or generating a realistic sample from scratch, which should be be aligned with the training set distribution, as if drawn for the same distribution.

In alignment with standard notation in the literature, the forward process is denoted by the distribution q, which remains fixed, while the reverse process is represented by p_θ, where θ denotes the parameters learned by a neural network. An illustrative overview of both processes is provided in Figure 3.1.

3.1.1 Forward Diffusion Process

Let q denote the distribution of real data, such that $\mathbf{x}_0 \sim q(\mathbf{x})$. The forward diffusion process is defined as a sequence of steps in which Gaussian noise is gradually added to the input data over T time steps. This yields a sequence of increasingly noisy samples $\mathbf{x}_0, \ldots, \mathbf{x}_T$, forming a Markov

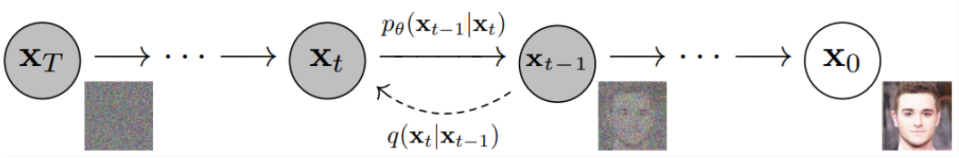

Figure 3.1: Image from *Denoising Diffusion Probabilistic Models* (Ho et al., 2020)

chain where each sample $\mathbf{x}_t$ depends only on its immediate predecessor. Thus, the transition satisfies the Markov property:

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

The entire forward process can then be described as:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \quad q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

Here, β_t represents a variance schedule with values in the interval $(0,1)$ and typically increases over time: $\beta_1 \leq \beta_2 \leq \cdots \leq \beta_T$.

While samples can be drawn from $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ recursively, it's also possible to directly condition on the original data point $\mathbf{x}_0$, obtaining samples from $q(\mathbf{x}_t|\mathbf{x}_0)$. We define the following terms: $\alpha_t = 1-\beta_t$, and the cumulative product $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$. Using a standardized form of the Gaussian distribution,

$$\mathbf{x} \sim \mathcal{N}(\mu, \sigma^2) \Rightarrow \mathbf{x} = \mu + \sigma\epsilon, \quad \text{where } \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}),$$

We can express the forward process iteratively as:

$$\begin{aligned}
\mathbf{x}_t &= \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\epsilon_{t-1} \\
&= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\epsilon_{t-1} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\bar{\epsilon}_{t-2} \\
&= \cdots \\
&= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_0
\end{aligned} \tag{3.1}$$

Here, $\bar{\epsilon}$ represents the effective noise term resulting from the sum of multiple Gaussian distributions. Using the identity

$$\mathcal{N}(\mathbf{0}, \sigma_1^2\mathbf{I}) + \mathcal{N}(\mathbf{0}, \sigma_2^2\mathbf{I}) = \mathcal{N}(\mathbf{0}, (\sigma_1^2+\sigma_2^2)\mathbf{I}),$$

the standard deviation becomes $\sqrt{(1-\alpha_t)+\alpha_t(1-\alpha_{t-1})} = \sqrt{1-\alpha_t\alpha_{t-1}}$.

As a result, we can express the marginal distribution of $\mathbf{x}_t$ conditioned on $\mathbf{x}_0$ at any arbitrary timestep t as:

$$q(\mathbf{x}_t|\mathbf{x}_0) \sim \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}) \tag{3.2}$$

Choice of Variance Schedule: The choice of variance schedule β_t plays a critical role in the behavior of the forward diffusion process, determining how rapidly noise is introduced. The original DDPM formulation used a simple linear schedule, linearly increasing β_t from $\beta_1 = 0.001$ to $\beta_T = 0.2$ [HJA20]. However, subsequent work has proposed alternative schedules to improve model performance.

One such alternative is the cosine schedule introduced in *Improved Denoising Diffusion Probabilistic Models* [ND21], which empirically showed improved results by reducing the amount of noise added during the early steps. The cosine schedule is defined as:

$$\bar{\alpha}_t = \frac{f(t)}{f(0)}, \quad f(t) = \cos\left(\frac{t/T + s}{1+s} \cdot \frac{\pi}{2}\right)^2$$

The variance terms β_t are then computed by:

$$\beta_t = 1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}},$$

with s being a small constant that offsets the cosine function. This schedule ensures that $\bar{\alpha}_t$ decreases smoothly from 1 to 0 as t approaches T.
It is crucial for the forward process that the final distribution approximates pure Gaussian noise. Specifically, we require:

$$q(\mathbf{x}_T) \approx \mathcal{N}(\mathbf{0}, \mathbf{I}),$$

which is confirmed by examining (3.2). As $\bar{\alpha}_T \approx 0$, the mean term vanishes, and so does the variance term approach 1, yielding:

$$q(\mathbf{x}_T|\mathbf{x}_0) \sim \mathcal{N}(\mathbf{x}_T; \sqrt{\bar{\alpha}_T}\mathbf{x}_0, (1-\bar{\alpha}_T)\mathbf{I}) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}) \tag{3.3}$$

Thus, in the limit as $T \to \infty$, the distribution converges to an isotropic Gaussian with zero mean and unit variance.

3.1.2 Reverse Diffusion process

In the reverse diffusion process, our goal is to model how to recover the original data features that were progressively obscured during the forward diffusion. To this end, we define the generative process as:

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t), \quad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$$

The distribution $p_\theta(\mathbf{x}_{0:T})$ describes the reverse diffusion process as a generative Markov chain, governed by the model parameters θ. Our primary interest lies in ensuring that the marginal distribution at the end of this chain, $p_\theta(\mathbf{x}_0)$, closely approximates the true data distribution $q(\mathbf{x}_0)$. Formally, this marginal is defined by integrating over all possible diffusion paths:

$$p_\theta(\mathbf{x}_0) := \int p_\theta(\mathbf{x}_{0:T})\, d\mathbf{x}_{1:T}$$

This marginal effectively aggregates the probability of all trajectories that could produce $\mathbf{x}_0$, and constitutes the target of our generative modeling.
To train the model, we aim to find parameters θ that maximize the likelihood of the observed data under the model, or equivalently, minimize the negative log-likelihood. However, computing this likelihood exactly is intractable, due to the high-dimensional integration over latent variables.

To address this challenge, we turn to variational inference — a technique for approximating complex probability distributions through optimization. The term "variational" stems from variational calculus, where the goal is to optimize functionals (i.e., functions of functions), and "inference" reflects our aim to infer the unknown model parameters. The key idea is to introduce a tractable approximate distribution, and frame learning as an optimization problem over this substitute tractable expression.

Within this framework, we use the Variational Lower Bound (VLB), also known as the Evidence Lower Bound (ELBO), as a substitute objective. This bound provides a lower approximation to the true log-likelihood. In practice, we minimize the negative VLB, which serves as an upper bound to the negative log-likelihood and aligns with standard loss minimization in machine learning:

$$-\log p_\theta(\mathbf{x}_0) \leq \mathbb{E}_q\left[\log \frac{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\right] =: L_{\text{VLB}}$$

This bound can be further expanded:

$$\begin{aligned}
L_{\text{VLB}} &= \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\log \frac{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\right] \\
&= \dots \\
&= \mathbb{E}_q\left[\underbrace{D_{\text{KL}}(q(\mathbf{x}_T \mid \mathbf{x}_0) \| p_\theta(\mathbf{x}_T))}_{L_T} + \sum_{t=2}^{T} \underbrace{D_{\text{KL}}(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \| p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t))}_{L_{t-1}} \underbrace{-\log p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1)}_{L_0}\right] \\
&= L_T + L_{T-1} + \dots + L_0
\end{aligned}$$

This decomposition gives us a sum of KL divergences between Gaussian distributions, each of which can be computed in closed form — a major advantage over relying on Monte Carlo estimates of the full log-likelihood. The full derivation of this is beyond the necessary complexity of this thesis, I will refer to Denoising Diffusion Probabilistic Models appendix A [HJA20]. As a result, we can efficiently train the model by minimizing these individual terms using stochastic gradient descent.

Concretely, the components of the VLB are:

$$\begin{aligned}
L_T &= D_{\text{KL}}(q(\mathbf{x}_T \mid \mathbf{x}_0) \| p_\theta(\mathbf{x}_T)) \\
L_{t-1} &= D_{\text{KL}}(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \| p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)) \\
L_0 &= -\log p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1)
\end{aligned}$$

In practice, we often disregard L_T during training. This is because $q(\mathbf{x}_T \mid \mathbf{x}_0)$ is a fixed Gaussian determined by the forward process, and $p_\theta(\mathbf{x}_T)$ is simply a standard normal distribution — both are non-learnable. Similarly, the final term L_0 is usually modeled with an independent decoder

that predicts $\mathbf{x}_0$ given $\mathbf{x}_1$ via a fixed Gaussian of the form $\mathcal{N}(\mathbf{x}_0; \mu_\theta(\mathbf{x}_1, 1), \sigma_1^2 \mathbf{I})$.

This leaves L_{t-1} as the primary term to optimize. It measures the divergence between the true posterior $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$ and the model prediction $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$. In the following section, we will explore in detail how to effectively learn this term, which forms the backbone of training for diffusion models.

3.1.3 Learning Objective

So far, we have gathered that the training objective in diffusion models is to minimize the KL divergence

$$D_{\mathrm{KL}}\left(q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) \| p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)\right).$$

To simplify this objective, we begin by expressing the true posterior using Bayes' rule:

$$\begin{aligned}
q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) &= q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{x}_0\right) \frac{q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_0\right)}{q\left(\mathbf{x}_t \mid \mathbf{x}_0\right)} \\
&\propto \exp\left(-\frac{1}{2}\left(\frac{\left(\mathbf{x}_t - \sqrt{\alpha_t}\mathbf{x}_{t-1}\right)^2}{\beta_t} + \frac{\left(\mathbf{x}_{t-1} - \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0\right)^2}{1-\bar{\alpha}_{t-1}} - \frac{\left(\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0\right)^2}{1-\bar{\alpha}_t}\right)\right) \\
&= \exp\left(-\frac{1}{2}\left(\frac{\mathbf{x}_t^2 - 2\sqrt{\alpha_t}\mathbf{x}_t\mathbf{x}_{t-1} + \alpha_t\mathbf{x}_{t-1}^2}{\beta_t} + \frac{\mathbf{x}_{t-1}^2 - 2\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0\mathbf{x}_{t-1} + \bar{\alpha}_{t-1}\mathbf{x}_0^2}{1-\bar{\alpha}_{t-1}} - \frac{\left(\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0\right)^2}{1-\bar{\alpha}_t}\right)\right) \\
&= \exp\left(-\frac{1}{2}\left(\left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right)\mathbf{x}_{t-1}^2 - \left(\frac{2\sqrt{\alpha_t}}{\beta_t}\mathbf{x}_t + \frac{2\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\mathbf{x}_0\right)\mathbf{x}_{t-1} + C\left(\mathbf{x}_t, \mathbf{x}_0\right)\right)\right)
\end{aligned}$$

Here, $C(\mathbf{x}_t, \mathbf{x}_0)$ is a function independent of $\mathbf{x}_{t-1}$, which we omit for brevity. The form of this expression reveals that it follows a Gaussian distribution, enabling us to extract the mean and variance using standard formulas. Remembering that $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$, we can define:

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

Using equation (3.2), we isolate $\mathbf{x}_0$ as:

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

and substituting this into $\tilde{\boldsymbol{\mu}}_t$ gives:

$$\begin{aligned}
\tilde{\boldsymbol{\mu}}_t &= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} \cdot \frac{1}{\sqrt{\bar{\alpha}_t}}\left(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t\right) \\
&= \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_t\right)
\end{aligned}$$

Thus, we arrive at the following expression for the true reverse distribution:

$$q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) \sim \mathcal{N}\left(\mathbf{x}_{t-1}; \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I}\right) = \mathcal{N}\left(\mathbf{x}_{t-1}; \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_t\right), \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t \mathbf{I}\right)$$

To define the model distribution $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$, we fix the variance to match the forward process schedule β_t, rather than learning it:

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

If two Gaussian distributions P and Q share the same known variance, then minimizing their KL divergence reduces to minimizing the squared error between their means:

$$\begin{aligned}\mathrm{KL}(P\|Q) &= \int \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu_1)^2}{2\sigma^2}} \log\left(\frac{e^{-\frac{(x-\mu_1)^2}{2\sigma^2}}}{e^{-\frac{(x-\mu_2)^2}{2\sigma^2}}}\right) dx \\ &\xrightarrow{\sigma_1=\sigma_2} = \frac{(\mu_1-\mu_2)^2}{2\sigma^2} \propto (\mu_1-\mu_2)^2\end{aligned}$$

This result also generalizes to multivariate Gaussian distributions.

Using this result, we can express the KL divergence loss L_{t-1} as a mean squared error between the true and predicted means:

$$\begin{aligned}L_{t-1} &= D_{\mathrm{KL}}\left(q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) \| p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)\right) \\ &\propto \left\|\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)\right\|^2\end{aligned}$$

Rather than predicting the mean directly, Ho et al. [HJA20] found it more effective to train the model to predict the noise component $\boldsymbol{\epsilon}$ instead. This leads to the following parameterization:

$$\begin{aligned}&\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) \\ &\Rightarrow \mathbf{x}_{t-1} \sim \mathcal{N}\left(\mathbf{x}_{t-1}; \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right), \beta_t \mathbf{I}\right)\end{aligned} \tag{3.4}$$

Note that this parameterization of $\mu_\theta(\mathbf{x}_t, t)$ is a deliberate modeling choice, and the variance in the learned distribution remains fixed at β_t, rather than using $\tilde{\beta}_t$. However, this has minimal impact, as $\tilde{\beta}_t \to \beta_t$ when $T \to \infty$.
The complete loss function is then written as:

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

Here, $\boldsymbol{\epsilon}_t$ is the noise used to corrupt $\mathbf{x}_0$ into $\mathbf{x}_t$, while $\boldsymbol{\epsilon}_\theta$ denotes the model's prediction of that same noise, based only on the noisy input $\mathbf{x}_t$.

3.1.4 Simplified Training

The authors of [HJA20] empirically observed that a simplified training objective yields better performance in practice. This alternative formulation omits the weighting terms present in the original objective, resulting in a more straightforward and stable loss function:

$$L_t^{\text{simple}} = \mathbb{E}_{\mathbf{x}_0,\epsilon_t}\left[\left\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t\right)\right\|^2\right]$$

This loss function directly trains the network to predict the noise component $\boldsymbol{\epsilon}_t$ added during the forward diffusion process, given a noised input at time step t. The training procedure can be summarized in the following algorithm:

Algorithm 1 Training

repeat
  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
  $t \sim \text{Uniform}(\{1,\dots,T\})$
  $\epsilon \sim \mathcal{N}(\mathbf{0},\mathbf{I})$
  Take gradient descent step on:
    $\nabla_\theta \left\|\epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t\right)\right\|^2$
until convergence

This algorithm, originally proposed by Ho et al., outlines the training loop for the denoising network $\boldsymbol{\epsilon}_\theta$. Given a data sample $\mathbf{x}_0$, a time step t is sampled uniformly at random, along with Gaussian noise ϵ. These are used to construct the noisy input $\mathbf{x}_t$ via the forward process, which is then passed through the network. The model is trained to recover the original noise ϵ, effectively learning to denoise $\mathbf{x}_t$ at arbitrary time steps.

3.2 Sampling

Sampling in this context refers to drawing samples from the learned model distribution $p_\theta(\mathbf{x}_0)$, which approximates the true data distribution $q(\mathbf{x}_0)$. Once the model $\epsilon_\theta(\mathbf{x}_t, t)$ has been trained to estimate the noise added during the forward process, we can use it to iteratively denoise a sample drawn from an isotropic Gaussian prior.

Starting from a sample $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0},\mathbf{I})$, we recursively apply the denoising process described in Equation (3.4), gradually removing noise over T steps to yield a sample from the model distribution. Each step approximates a sample from $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, effectively reversing the diffusion process and transforming pure noise into a realistic data point.

The complete sampling procedure is outlined in Algorithm 2:

Algorithm 2 Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0},\mathbf{I})$
for $t = T,\dots,1$ **do**
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right) + \sigma_t\mathbf{z}$
end for
return $\mathbf{x}_0$

This backward process, guided by the trained network, incrementally reduces noise in the sample, ultimately producing a clean output $\mathbf{x}_0$ that resembles the original data distribution.

3.3 Hijacking the Sampling Process

In the standard formulation of Denoising Diffusion Probabilistic Models (DDPMs) as we have seem so far, the reverse sampling process iteratively generating samples from a learned conditional distribution $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$. This distribution is modeled as a Gaussian whose mean is determined by the output of a neural network (predicting the noise added in the forward process), and whose variance is set by a pre-defined schedule. Sampling from this distribution takes the form:

$$\mathbf{x}_{t-1} = \mu + \sqrt{\beta_t}\mathbf{z}$$
$$\text{Where:} \quad \mu = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right)$$

where $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$ is isotropic Gaussian noise. This noise is essential to capture the stochastic nature of the generative process. Without it, sampling becomes deterministic and reduces the diversity of generated outputs, leading to the theory of Denoising Diffusion Implicit Models [SME20]. However, this noise term is completely unconditioned: it assumes no structure in the underlying data, and treats all directions in the sample space equally.

Yet, in many domains, we possess strong priors or ground truths about the structure of the data we aim to generate. For instance, in image upscaling, the average of high-resolution pixel blocks must match known low-resolution pixels; in physical simulations, local conservation laws (e.g., of mass or concentration) must hold. These constraints are often inherently present in the dataset. Thus, the trained model output is expected to implicitly respect them through ϵ_θ. However, the added variance noise $\mathbf{z}$ can act in opposition, perturbing samples in ways that violate these properties.

The key insight is that while the model prediction ϵ_θ may already respect structural constraints through training, the sampling noise $\mathbf{z}$ does not, and this opens the door for improvement. We propose a principled method to *hijack* the sampling process by modifying the way $\mathbf{z}$ is drawn at each timestep. Rather than sampling it independently for each dimension, we draw from a carefully constructed multivariate Gaussian distribution designed to preserve specified constraints in the resulting sample $\mathbf{x}_{t-1}$. This way we still adhere to the Markovian nature of the diffusion model, whilst keeping certain aspects of generated distribution fixed.

3.3.1 Conditional Multivariate Sampling

For a general constraint where we require a set of variables to satisfy a linear relation, we can modify the noise sampling process using principles from conditional multivariate Gaussian distributions. Let us consider the general framework from multivariate statistics for conditioning on a subset of variables, following litterature of the wiki [Wik25] under conditional distributions.

Given an N-dimensional random vector $\mathbf{x}$ drawn from a multivariate normal distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, we can partition it as:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} \text{ with sizes } \begin{bmatrix} q \times 1 \\ (N-q) \times 1 \end{bmatrix}$$

Similarly, the mean vector and covariance matrix can be partitioned as:

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix} \text{ with sizes } \begin{bmatrix} q \times 1 \\ (N-q) \times 1 \end{bmatrix}$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix} \text{ with sizes } \begin{bmatrix} q \times q & q \times (N-q) \\ (N-q) \times q & (N-q) \times (N-q) \end{bmatrix}$$

When we condition on $\mathbf{x}_2 = \mathbf{a}$, the distribution of $\mathbf{x}_1$ becomes:

$$(\mathbf{x}_1 | \mathbf{x}_2 = \mathbf{a}) \sim \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$$

where:

$$\boldsymbol{\mu}_c = \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\mathbf{a} - \boldsymbol{\mu}_2) \tag{3.5}$$

$$\boldsymbol{\Sigma}_c = \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21} \tag{3.6}$$

Average Condition: We can apply this framework to the general case where we want to enforce a constraint on the average of a set of variables. Let us denote n variables as $x_1, x_2, \ldots, x_n$ with corresponding means $\mu_1, \mu_2, \ldots, \mu_n$ and variance σ^2. Our objective is to sample these variables such that:

$$\frac{1}{n}\sum_{i=1}^{n} x_i = A$$

To apply the conditional multivariate Gaussian framework, we define:

- $\mathbf{x}_1 = (x_1, x_2, \ldots, x_{n-1})$, the first $n-1$ variables
- $\mathbf{x}_2 = \frac{1}{n}\sum_{i=1}^{n} x_i$, the average of all n variables
- $\mathbf{a} = A$, our target average value

We then construct the joint distribution of $(x_1, x_2, \ldots, x_{n-1}, a)$ with:

$$\boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_{n-1} \\ \frac{1}{n}(\mu_1 + \mu_2 + \cdots + \mu_n) \end{pmatrix}$$

For the covariance structure, with the diffusion variance scheduled of identity covariance $\sigma^2\mathbf{I}$, we derive:

$$\Sigma = \begin{bmatrix} \mathbf{\Sigma}_{11} & \mathbf{\Sigma}_{12} \\ \mathbf{\Sigma}_{21} & \mathbf{\Sigma}_{22} \end{bmatrix} = \sigma^2 \begin{bmatrix} \mathbf{I}_{n-1} & \frac{1}{n}\mathbf{1}_{n-1} \\ \frac{1}{n}\mathbf{1}_{n-1}^T & \frac{1}{n} \end{bmatrix}$$

where $\mathbf{1}_{n-1}$ is a vector of ones of length $n-1$. Applying the conditional distribution formulas from equation 3.5, 3.6, we get:

$$\boldsymbol{\mu}_c = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_{n-1} \end{pmatrix} + \frac{nA - (\mu_1 + \mu_2 + \cdots + \mu_n)}{n} \mathbf{1}_{n-1}$$

$$\mathbf{\Sigma}_c = \sigma^2 \mathbf{I}_{n-1} - \frac{\sigma^2}{n^2} \cdot n \cdot \mathbf{1}_{n-1}\mathbf{1}_{n-1}^T = \sigma^2 \left(\mathbf{I}_{n-1} - \frac{1}{n}\mathbf{J}_{n-1} \right)$$

where $\mathbf{J}_{n-1}$ is the $(n-1) \times (n-1)$ matrix of all ones.
This gives us the distribution from which to sample the first $n-1$ variables. Once these have been sampled, the n-th variable is deterministically computed as:

$$x_n = nA - \sum_{i=1}^{n-1} x_i$$

It is worth noting that this approach generalizes to other tractable linear constraints beyond averages. For example, if we wanted to constrain the sum of variables rather than their average, we would adjust the definitions of $\mathbf{x}_2$ and $\mathbf{a}$ in our framework, resulting in different formulations for $\mathbf{\Sigma}_{12}$, $\mathbf{\Sigma}_{21}$, and $\mathbf{\Sigma}_{22}$. The fundamental principle, conditioning a multivariate Gaussian on a linear constraint, remains the same across different types of constraints.

This conditional sampling approach ensures that our variables are sampled in a statistically coherent manner that respects the specified constraint, while maintaining as much of the original variability as possible. This is particularly valuable in the context of diffusion models, where we want to preserve the stochastic nature of the process while enforcing structural properties in the generated outputs.

3.3.2 Use Case: Image Upscaling

An example of such a constraint-driven problem is image upscaling. When increasing the resolution of an image (i.e. 64×64 to 128×128), we often have access to the original low-resolution image. A natural constraint is that each pixel in the low-resolution image should equal the average of the corresponding block of pixels in the high-resolution image.

Formally, let $\mathbf{x}_0^{\text{low}} \in \mathbb{R}^{H \times W}$ denote the known low-resolution image, and let $\mathbf{x}_0^{\text{high}} \in \mathbb{R}^{2H \times 2W}$ be the high-resolution image to be generated. We define the constraint:

$$\mathbf{x}_0^{\text{low}}[i,j] = \frac{1}{4} \sum_{(u,v) \in \mathcal{N}(i,j)} \mathbf{x}_0^{\text{high}}[u,v]$$

where $\mathcal{N}(i,j)$ denotes the 2×2 patch in the high-resolution image corresponding to pixel (i,j) in the low-resolution image. To satisfy this constraint, we apply the method like we have explained

above.

For each 2×2 patch, and for each timestep t in the diffusion sampling process: We get ϵ_θ from the model, and by extension μ_θ. Instead of Sampling the values $\mathbf{z}$ from isotropic noise to get $\mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \beta_t \mathbf{I})$. We make the above assumption on the last value, to improve the Gaussian to draw from the conditional distribution $\mathcal{N}(\mathbf{x}_{t-1}; \mu_c(\mathbf{x}_t, t), \beta_t \Sigma_c)$. The deduced values are for $n = 4$:

$$\boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \frac{1}{4}(\mu_1 + \mu_2 + \mu_3 + \mu_4) \end{pmatrix}, \quad \boldsymbol{\Sigma} = \sigma^2 \begin{pmatrix} 1 & 0 & 0 & 1/4 \\ 0 & 1 & 0 & 1/4 \\ 0 & 0 & 1 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 \end{pmatrix}$$

With equation 3.5 and 3.6 we find:

$$\boldsymbol{\mu}_c = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{pmatrix} + A - \frac{(\mu_1 + \mu_2 + \mu_3 + \mu_4)}{4}, \quad \boldsymbol{\Sigma}_c = \sigma_t^2 \left[\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \frac{1}{4} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \right] = \sigma_t^2 \left[\boldsymbol{I}_3 - \frac{1}{4} \boldsymbol{J}_3 \right]$$

Where A is the equivalent pixel in x_0^{low} that determines what the average should be. Then lastly we would calculate the last pixel deterministically. The algorithm can be viewed in algorithm 3

Algorithm 3 Constraint-Aware Sampling with Correlated Noise for Upscaling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \ldots, 1$ **do**
  $\boldsymbol{\mu} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right)$
  $\boldsymbol{\Sigma}_c = \sigma_t^2 \left[\boldsymbol{I}_3 - \frac{1}{4} \boldsymbol{J}_3 \right]$
  for each 2×2 patch (i, j) in the image **do**
    $\mu_c^{(i)} = \mu^{(i)} + \mathbf{x}_0^{\text{low}}[i, j] - \frac{(\mu^{(1)} + \mu^{(2)} + \mu^{(3)} + \mu^{(4)})}{4}, \quad i \in \{1, 2, 3\}$
    $\boldsymbol{z}^i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_c), \quad i \in \{1, 2, 3\}$
    $\boldsymbol{x}_{t-1}^i = \boldsymbol{\mu}_c^{(i)} + \boldsymbol{z}_i, \quad i \in \{1, 2, 3\}$
    $\boldsymbol{x}_{t-1}^{(4)} = 4 \cdot \mathbf{x}_0^{\text{low}}[i, j] - \sum_{i=1}^{3} \boldsymbol{x}_{t-1}^{(i)}$
  end for
end for
return $\mathbf{x}_0^{\text{high}}$

This strategy not only maintains the strict mathematical constraint between low- and high-resolution representations throughout the denoising process, but also ensures that the noise is sampled in a way that preserves the statistical properties expected from a diffusion model. The correlated noise structure prevents any single pixel from bearing disproportionate responsibility for maintaining the constraint, leading to more natural, high-quality upscaled images. The negative correlation in the covariance matrix subtly encourages diversity within each patch while ensuring collective adherence to the constraint.

This approach strikes an ideal balance: it preserves the expressive power of diffusion models while constraining the solution space to those samples which are structurally valid, without requiring any retraining or architectural changes to the underlying model.

3.3.3 Use Case: Concentration Field Conservation

In this use case, the image represents a concentration field, and we require that the global average concentration across all pixels equals 1. Unlike the upscaling scenario where constraints apply locally, this constraint spans the entire image and must be satisfied exactly throughout the denoising process.

The global average constraint is a linear condition and can be handled using the conditional multivariate Gaussian framework following the same procedure as the upscaling case. Let $\mathbf{x}_t \in \mathbb{R}^n$ be the current noisy image at timestep t, where n is the total number of pixels. The diffusion model predicts the mean $\boldsymbol{\mu} \in \mathbb{R}^n$ for the reverse process. We want to sample $\mathbf{x}_{t-1}$ such that the average of its entries equals 1:

$$\frac{1}{n}\sum_{i=1}^{n} x_{t-1}^{(i)} = 1$$

From the theoretical framework developed earlier, the conditional distribution of $\mathbf{x}_{t-1}$ given this constraint is a multivariate Gaussian with adjusted mean and covariance:

$$\boldsymbol{\mu}_c = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_{n-1} \end{pmatrix} + \left(1 - \frac{1}{n}\sum_{i=1}^{n} \mu^{(i)}\right) \mathbf{1}_{n-1}$$

$$\boldsymbol{\Sigma}_c = \sigma_t^2 \left(\mathbf{I}_{n-1} - \frac{1}{n}\mathbf{J}_{n-1}\right)$$

While this formulation is theoretically sound, directly computing and sampling from this covariance matrix would be computationally prohibitive for typical image sizes (e.g., $n = 128 \times 128 = 16,384$ in our case, but grow too large for larger images). The algorithm for this is similar to algorithm 3, but without the patch based for loop.

3.3.4 Use Case: 1D Brownian Bridge

In this case, the generated signal is required to form a 1D Brownian bridge. Rather than training the diffusion model to generate the full Brownian bridge directly, we instead train it to generate the increments of the process. The cumulative sum of these increments then yields the Brownian bridge trajectory. This framing transforms the global constraint (fixed start and end values) into a linear constraint on the sum of increments, and by extension the average of increments, which is ideally suited to the global conditional Gaussian procedure described in concentration conservation case.

This approach offers several benefits. First, it enforces the Brownian bridge structure exactly during sampling. Second, it retains the expressive power and generative diversity of the diffusion model without retraining. Finally, by working in increment space, we maintain compatibility with the linear constraint framework, allowing efficient sampling via conditional Gaussians. As

a result, we generate samples that are not only statistically valid but also adhere strictly to the boundary conditions of a Brownian bridge.

3.4 Architecture

Up to this point, we have discussed the theoretical aspects of the diffusion process, but not the neural network architecture used to implement the denoising function ϵ_θ. One of the strengths of Denoising Diffusion Probabilistic Models (DDPMs) is that they are relatively agnostic to the choice of network architecture. The primary constraint is that the input and output of the network must have the same shape, as the model operates in an iterative loop where each output becomes the input for the next time step.

Popular architectural choices include Residual Networks (ResNets) [He+15] and U-Nets [RFB15], both of which can maintain consistent input-output dimensions and have proven effective in generative tasks.

Validation of Generative AI

4

The validation of generative models remains an evolving and complex challenge within the field of machine learning. Despite significant advances, there is no universally accepted method for evaluating generative models, as highlighted by the paper A Study on the Evaluation of Generative Models [Bet+22]. This challenge stems from the nature of generative models, such as GANs and Diffusion models, which aim to generate diverse outputs that reflect the statistical properties of the training data distribution, rather than producing exact replicas of the training images.

The primary goal in validating generative models is to ensure that the generated images come from the same distribution as the real images. However, the generated samples should ideally show variation while still resembling the characteristics of the original data. This distinction is crucial: a generative model should not simply copy images but should capture the underlying distribution from which the real data is drawn.

For general validation, one intuitive method is to visually inspect the generated images to assess whether they resemble samples from the target distribution. For instance, when generating images of cats, it is relatively straightforward for human observers to determine if the generated images belong to the same distribution as the training set. However, in more specialized or unfamiliar domains, such as generating images based on abstract scientific data like fluid dynamics concentration fields, or Brownian Bridges. In such cases, visual inspection by humans may not be feasible, and quantitative metrics become necessary to assess the quality of the generated samples.

In the following sections, I will introduce several common metrics used across various problem domains in generative modeling. Each metric comes with its own strengths and limitations, serving specific evaluation purposes. Later, in the experiments chapter, I will explain how we adapt these metrics to suit the different use cases, ensuring the most meaningful and accurate assessment of our results.

4.1 Inception Score

The **Inception Score (IS)**, proposed in the paper Improved Techniques for Training GANs [Sal+16], is one of the most commonly used metrics for validating generative models. The IS uses a pre-trained classification network (such as InceptionV3) to evaluate two main aspects of the generated images: (1) the distinctiveness of the generated image's predicted class, and (2) the diversity of the generated samples. The IS is calculated as follows:

$$IS = \exp\left(\mathbb{E}x \sim p_G[KL(p_\theta(y|x)||p_\theta(y))]\right)$$

Where $p_\theta(y|x)$ represents the classification distribution of the image. and $p_\theta(y)$ is the distribution of labels across the dataset. The goal is to measure how clear and distinctive the output class is, as well as the diversity of the generated images.

While IS is a widely recognized metric, it has certain limitations. First, it assumes that the generated images can be classified into discrete categories, which may not apply to more complex, continuous domains. Additionally, since InceptionV3 is trained on the ImageNet dataset, it may not yield meaningful results for specialized tasks, such as generating any of the task that we have defined. For these reasons, this paper does not rely on the Inception Score for evaluation, in line with the recommendations of [Bet+22].

4.2 Fréchet Inception Distance

The **Fréchet Inception Distance (FID)**, introduced in [Heu+18], is another widely used metric for assessing generative models. Unlike the IS, which evaluates the final generated image distribution, FID operates on the latent space distribution of a pre-trained neural network. The goal of FID is to compare the generated data's distribution in latent space to that of the real data. It is typically computed using the InceptionV3 network, but like IS, it can be adapted for other models if necessary. The FID score is calculated as:

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

where μ_r and Σ_r are the mean and covariance of the real data's latent space representations, and μ_g and Σ_g are the corresponding statistics for the generated data. The metric assumes that these latent representations follow a multivariate Gaussian distribution, and measures how closely the distribution of generated data matches the real data in feature space.

Although FID has become the standard metric for evaluating generative models, it has its own drawbacks. The assumption of a Gaussian distribution in latent space may not hold for all types of data, and studies on the matter have shown that FID can be biased [CF20]. Despite these limitations, FID remains a valuable tool, especially when the model's latent space exhibits Gaussian-like behavior. Relaxing this Gaussian assumption, has also been attempted leading to the metric Kernel Inception Distance (KID) [Bińkowski+21]. I will hover not be using this metric over FID, as in the space of research of diffusion generative model, we see little to no use, comparatively to FID.

4.3 Peak Signal-to-Noise Ratio (PSNR)

The **Peak Signal-to-Noise Ratio (PSNR)** is one of the most widely used traditional metrics for evaluating image quality, particularly in tasks such as image compression, denoising, and reconstruction. PSNR quantifies the similarity between a reference image and a generated or processed image by measuring the logarithmic ratio between the maximum possible pixel value and the mean squared error (MSE) between the two images. The PSNR is defined as:

$$PSNR = 10 \cdot \log_{10} \left(\frac{MAX_I^2}{MSE} \right)$$

where:

- MAX_I is the maximum possible pixel value of the image (e.g., 255 for 8-bit images),
- MSE is the mean squared error between the reference and generated images:

$$MSE = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} [x(i,j) - y(i,j)]^2$$

PSNR is expressed in decibels (dB), with higher values indicating greater similarity between the images. A PSNR value above 30 dB is generally considered to indicate good reconstruction quality, although acceptable thresholds may vary depending on the application and image content.

Despite its simplicity and interpretability, PSNR has notable limitations. It operates on a purely pixel-wise basis and does not account for perceptual factors such as texture, structure, or contextual semantics. As a result, two images may yield high PSNR values while appearing visually dissimilar to a human observer. This is especially problematic in generative modeling tasks, where perceptual quality and structural coherence often outweigh exact pixel-level fidelity.

In the context of generative models such as GANs and Diffusion models, PSNR is only relevant when there exists a clear ground-truth reference image, such as in image-to-image translation or super-resolution tasks. However, in settings where the goal is to sample novel images from a learned data distribution rather than replicate specific targets, the utility of PSNR becomes limited. For this reason, while PSNR can be a useful supplementary metric for quantifying low-level distortions, it should not be relied upon in isolation for evaluating the fidelity or realism of generative outputs.

4.4 Structural Similarity Index (SSIM) for Image Quality Assessment

Structural Similarity Index (SSIM) is a perceptual metric introduced by Wang et al. [Wan+04] to assess the similarity between two images by explicitly modeling three aspects of visual perception: luminance, contrast, and structure. Unlike traditional metrics such as Mean Squared Error (MSE), which only quantify pixel-wise differences, SSIM aims to capture changes that are more relevant to human observers by comparing local patterns of pixel intensities after normalizing for these three perceptual components

The SSIM index is constructed by first defining three comparison functions:

- **Luminance:** Measures the similarity of the mean intensity between the images.
- **Contrast:** Compares the standard deviations, reflecting contrast differences.
- **Structure:** Evaluates the correlation (covariance) of the normalized images, capturing structural similarity.

Mathematically, these are expressed as:

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}$$
$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}$$
$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}$$

where μ_x and μ_y are the mean pixel values, σ_x and σ_y are the standard deviations, σ_{xy} is the covariance, and C_1, C_2, C_3 are small constants to stabilize the division.

The general form of SSIM combines these components multiplicatively, often with exponents set to 1 for simplicity:

$$SSIM(x,y) = l(x,y)^\alpha c(x,y)^\beta s(x,y)^\gamma$$

Setting $\alpha = \beta = \gamma = 1$ and $C_3 = C_2/2$, the formula simplifies to the widely used version:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

where:

- μ_x, μ_y: Mean pixel values of images x and y
- σ_x^2, σ_y^2: Variances of x and y
- σ_{xy}: Covariance between x and y
- C_1, C_2: Constants for numerical stability (typically $C_1 = 0.01$, $C_2 = 0.03$)

The SSIM score ranges from 0 to 1, where 1 indicates perfect perceptual similarity. This makes SSIM especially valuable in image processing tasks such as super-resolution and upscaling, where preserving structural information and perceptual quality is more important than pixel-wise accuracy.

5 Experiments

This chapter evaluates the proposed method of constraint-aware sampling across three distinct use cases, each deliberately chosen to test the method on a diverse problem space. These cases represent different types of structural constraints: spatial averaging in image super-resolution, global conservation laws in fluid concentration fields, and boundary conditioning in one-dimensional stochastic processes. While the underlying diffusion model architecture and training procedures remain consistent in most cases, the sampling strategy is uniquely adapted to enforce the relevant constraint in each task. We present each scenario with details on dataset, training setup, conditional sampling approach, and validation strategy.

It should be noted, that all specific implementation details can be viewed in the following git repository [Smo].

5.1 Use Case: Concentration Field Conservation

In the first experiment, we aim to investigate whether the proposed conditional sampling technique can enforce global physical constraints, specifically, conservation laws, during the generative process of a diffusion model. For this purpose, we use data from the *Polymathic Active Matter* dataset, published as part of `the well` dataset collection on physical simulations [AI25].

The active matter dataset simulates the behavior of rod-like particles suspended in a two-dimensional viscous fluid. The underlying model is a continuum kinetic theory, capturing the effects of internal stresses, particle alignment, and fluid interactions. Each simulation yields a spatiotemporal sequence consisting of 81 frames over 20 seconds on a 256×256 spatial grid. The dataset contains four core physical quantities: a scalar concentration field, a 2D velocity vector field, and tensor fields representing orientation and strain rate.

For our purposes, we focus exclusively on the scalar *concentration field The Wells* Active Matter simulation. We downsample each simulation frame to a 128×128 resolution to reduce training complexity and memory usage. Importantly, the concentration field in each frame is governed by a conservation law: the mean concentration must always sum to a constant value, specifically 1.0. This serves as a natural, global constraint that we will aim to preserve through conditional sampling.

To introduce diversity in the training data and generalize over a range of system behaviors, we train the model on a wide array of simulation instances generated using randomly sampled physical parameters ξ and α, which influence alignment strength and activity levels in the underlying model.

5.1.1 Training Procedure

The model architecture used for this task is a mid-sized U-Net from the `diffusers` library. The model is configured for grayscale scalar inputs with one channel. The noise scheduler follows the DDPMScheduler with 1000 diffusion steps and no sample clipping, because we are not trying to generate typical images in $[-1, 1]$, which the diffusers library is designed for by default. The optimizer used is AdamW with a cosine learning rate scheduler and warmup phase. Details of the specific structure can be viewed in 10.2, or in the code in the Github.

The training set comprises 14,000 unique simulation frames and is trained for 500 epochs using standard MSE loss between the predicted noise ϵ_θ and the target Gaussian noise added during the forward process. No constraints are introduced during training, allowing us to isolate the effects of the proposed method during inference only.

5.1.2 Conditional Sampling Method

During sampling, we aim to enforce the conservation constraint that the average pixel value of the generated concentration field equals 1.0. This is a global, tractable constraint that can be implemented directly in the sampling process by modifying the noise term $\mathbf{z}$ at each timestep.

Instead of sampling isotropic Gaussian noise $\mathbf{z} \sim \mathcal{N}(0, I)$ independently for all pixels, we treat one pixel per image (or a set of pixels) as dependent on the others. More concretely, we randomly sample all but one pixel from the standard normal distribution and analytically solve for the remaining pixel such that the constraint $\frac{1}{N^2}\sum_{i,j} x_{ij}^{(t)} = 1$ holds at each timestep. This adjustment is propagated through the sampling equation:

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right) + \sqrt{\beta_t}\,\mathbf{z}_c$$

Here, $\mathbf{z}_c$ denotes the constrained noise vector, which ensures the conservation condition is satisfied at each step t. This small but targeted intervention is designed to steer the generative process toward physically valid outputs, without requiring architectural changes or retraining.

5.1.3 Validation Setup

The validation set is at ≈ 2000 instances. Because we do not aim to reconstruct any ground truth images, but rather we synthesize arbitrary animation instances, that should only have the attribute; that the image generated should be from the true data distribution, for which the training data is in of itself only a subset of the true distribution. Standard pixel-wise loss metrics such as MSE are not applicable. Instead, we evaluate the realism and fidelity of generated samples using distributional and perceptual metrics.

We primarily rely on the Fréchet Inception Distance (FID), which we optimally would like adapted to this domain, by training a lightweight convolutional neural network to predict the underlying physical parameters ξ, from the generated frames. The feature activations of this domain-specific network would serve as the embedding space for the FID calculation. This approach would have ensured that the metric is aligned with the relevant physical dynamics, unlike standard FID

models trained on object-centric image datasets.

Unfortunately this approach proved not feasible, as the simulation parameters doesn't seem to contain necessary spatial information that allows for a CNN to be trained. More so, these parameters are perhaps more present in the temporal space of the animation. For this reason, we resort to the standard inception model `InceptionV3` which have been endorsed in prior literature for evaluating generative quality [Bet+22], these models are ultimately trained on object-centric image datasets and make assumptions that diverge significantly from the nature of our data. But despite this, we are not particularly concerned with the absolute value of the metrics in this thesis. Instead, our concern is that the metrics can outperform each other for the different models, whilst arguably having uniform theoretical ground for the models performance. Because whether or not the Inception model is trained on the specific model. We still have a distribution in the inception model from the true data and from arbitrary data, for which it would still be true, that if the generated data *is indeed drawn from the true distribution*, then the inception model values would show a similar distribution.

Even so, because of the critical aspects of the above statement, I will in addition to FID, report the Structural Similarity Index Measure (SSIM) and Peak-Signal-to-Noise Ratio (PSNR) between generated samples and the best matching sample in the validation set. While SSIM and PSNR are traditionally designed for paired comparisons, such as in image compression or reconstruction, this nearest-neighbor strategy allows us to repurpose them as statistical indicators of realism. The underlying assumption is that a better generative model will, in expectation, produce samples that lie closer to the support of the real data distribution. Thus, for a sufficiently large number of generated samples and validation images, a model that captures more of the true data distribution will achieve higher average SSIM and PSNR scores under this best-match comparison strategy. We emphasize that these metrics are not used to assess whether individual images are accurately reconstructed, but rather to measure how structurally similar the generated outputs are to any plausible real instance. which should under sufficiently large data generation also indicate a superiority of model performance.

Another reason for using SSIM and PSNR, even though their use is questionable; is that we can use these metrics to show that generated images might/is be present in the true data distribution. This introduces a human level subjective evaluation, which should not be undervalued necessarily, if not it can bring nuance to the discussion.

5.2 Use Case: Super-Resolution

In this experiment, we evaluate the proposed conditional sampling method on a classical super-resolution task, where the goal is to reconstruct a high-resolution image from its downsampled counterpart. This problem presents a straightforward and naturally occurring constraint: each pixel in the low-resolution image is the arithmetic mean of a corresponding 2×2 patch in the high-resolution image. Our method leverages this prior to enforce consistency between the generated high-resolution image and the known low-resolution input.

We use the `huggan/smithsonian_butterflies_subset` dataset, which contains photographs of butterflies against predominantly white backgrounds. The dataset is not selected for complexity or diversity, but rather for its simplicity and repeated use. Each high-resolution image is resized to 64×64 pixels, while the corresponding low-resolution version is created by downsampling to 32×32 pixels.

5.2.1 Training Setup

To maintain consistency across experiments and isolate the effects of the conditional sampling procedure, we use the same model architecture and training hyperparameters as in the concentration field experiment, with the only change that the model should accept 3 channel input and output data. The model is trained for 200 epochs on a reduced subset of 1,000 butterfly images (≈ 800). We train the model to produce arbitrary images of butterflies, and is then later in the sampling process introduced to the low res images only.

5.2.2 Conditional Sampling

During inference, we apply the constraint-aware sampling method as outlined in Section 3.3.2. Specifically, the sampling noise $\mathbf{z}$ is no longer fully isotropic; instead, for each 2×2 patch of pixels, three pixel values are sampled normally, and the fourth is deterministically computed to ensure the local average equals the known low-resolution value. This procedure is summarized in Algorithm 3.

- **Baseline (Unconstrained) Sampling:** Instead of initializing the denoising chain from pure noise, we begin from a noised version of the low-resolution image at some intermediate timestep (e.g., $t = 100$). The model then denoises from this state, using the same learned process as in training. This method is similar to DDIM-like techniques where sampling starts mid-way, and has been shown to produce coherent results when used with partial information.
- **Conditional Sampling:** As described above, the generative process is modified at each timestep to enforce the 2×2 averaging constraint directly in the noise sampling, thereby guiding the model toward structurally consistent outputs.

5.2.3 Validation Strategy

Although the model is trained to generate arbitrary high-resolution butterfly images, our goal during inference is to upscale a given low-resolution image. In both the baseline and conditional sampling pipelines, the low-resolution image serves as an anchor to guide the generative process.

Because the low-resolution image corresponds to a ground truth high-resolution image, we are able to evaluate output fidelity directly. We employ the Mean Squared Error (MSE) as the primary metric for quantitative evaluation, along with the Structural Similarity Index (SSIM) and Signal-to-Noise-Ratio (SNR) to capture perceptual quality, which in this case is a more appropriate use of the metrics. While MSE is a straightforward pixel-wise difference metric, SSIM accounts for local structural features, which can be especially useful for images with subtle

textures like butterfly wings.

5.3 Use Case: 1D Brownian Bridge

5.3.1 Dataset and Task Description

In this final experiment, we investigate how our constraint-aware sampling approach performs on a simple one-dimensional stochastic process: the Brownian bridge. This process is a constrained version of Brownian motion in which a random trajectory starts at an initial point and is conditioned to end at a fixed final value.

For our purposes, we generate a synthetic dataset of Brownian bridge trajectories, where each sample begins at zero and is constrained to terminate at zero also. Each trajectory consists of 64 evenly spaced timesteps, and the intermediate values are generated to reflect natural Brownian fluctuation while satisfying the endpoint condition. This setup offers a minimal, well-understood testbed for evaluating constraint enforcement in a low-dimensional setting.

5.3.2 Training Setup

Unlike the previous image-based use cases, this experiment requires a custom 1D U-Net model, as the `diffusers` library currently 1D diffusion model is currently not functioning.

> The `diffusers` library, does support 1D duffusion models. But is currently not working, as i had to discover the hard way; *[Hugging Face Community Thread Link]*

Because of this we require a custom 1D Diffusion model to be built. The architecture consists of three downsampling blocks and three upsampling blocks, with channel dimensions progressing through 32, 64, and 128 of a standard U-net with sinusoidal time embedding. The model arcitecture is closely similar to the first introduction from [HJA20], with time embedding originally from position embedding of NLP [Vas+17].

The model is trained on a dataset of one million Brownian bridge trajectories for 200 epochs using a standard DDPM noise scheduler with 1000 diffusion steps. Importantly the training is not on the bridges, but on the jumps at each steps. Meaning that the complete Brownian Bridge can be found by cumulated sum over the model output.

5.3.3 Conditional Sampling

We output the jumps of the Brownian Bridge because the conditional sampling method is more meaningful, and can thereby be implemented similar to the other statements as averages of the samples. At inference time, we enforce the endpoint constraint using the method described in Section 3.3.4. In short, we modify the sampling procedure to ensure that the final value of each generated trajectory exactly equals the fixed target value, while still preserving the stochasticity of the intermediate steps.

5.3.4 Validation Strategy

Validation in this setting poses a different challenge compared to the image-based tasks. Since our model is trained to generate diverse trajectories rather than match any specific ground truth sequence, pointwise comparison metrics like MSE or SSIM are inappropriate. Instead, the goal is to ensure that the generated trajectories collectively resemble the statistical properties of the Brownian bridge distribution. FID is also inappropriate for all the same reasons as the active matter case, but in addition we generate 1D data, further diverging from the inception model. Instead of these metrics, we leverage known analytical properties of Brownian bridges for validation. For a Brownian bridge $B(t)$ defined on $[0,1]$ with $B(0) = B(1) = 0$, we can compare the empirical distributions of key random variables derived from the generated trajectories against their known analytical distributions. Specifically, we focus on two quantities:

1. The supremum of the absolute value of the bridge:

$$K = \sup_{t\in[0,1]} |B(t)|,$$

   whose cumulative distribution function (CDF) is given by:

$$\Pr(K \leq x) = 1 - 2\sum_{k=1}^{\infty}(-1)^{k-1}e^{-2k^2x^2} = \frac{\sqrt{2\pi}}{x}\sum_{k=1}^{\infty}e^{-(2k-1)^2\pi^2/(8x^2)}.$$

2. The joint distribution of the last zero-crossing before the midpoint (τ_-) and the first zero-crossing after the midpoint (τ_+), which decomposes the bridge into smaller segments. The joint probability density function (PDF) of τ_- and τ_+ is:

$$\rho(\tau_-, \tau_+) = \frac{1}{2\pi\sqrt{\tau_-(1-\tau_+)(\tau_+ - \tau_-)^3}}.$$

   These can be sampled conditionally as:

$$\tau_+ = \frac{1}{1+\sin^2\left(\frac{\pi}{2}U_1\right)} \in \left(\frac{1}{2}, 1\right), \quad \tau_- = \frac{U_2^2\tau_+}{2\tau_+ + U_2^2 - 1} \in \left(0, \frac{1}{2}\right),$$

   where U_1, U_2 are uniformly distributed random variables over $(0,1)$.

To validate the generated data, we compute the empirical distributions of K, $\rho(\tau_-, \tau_+)$ from the generated trajectories and compare them to their analytical counterparts using the Kullback-Leibler (KL) divergence. A small KL divergence indicates that the generated data closely matches the statistical properties of true Brownian bridges, ensuring the fidelity of our model. This approach provides a rigorous quantitative measure of the quality of the generated trajectories without relying on pointwise comparisons or image-based metrics.

Results

6

In this chapter, we present the results from the different experimental setups. Each experiment compares unconditional and conditional generative diffusion models, evaluated using a range of quantitative metrics and qualitative visualizations.

6.1 Image Upsampling – Butterfly Dataset

Quantitative results for the image upsampling task on the butterfly dataset are shown in Table 6.1. The conditional model consistently outperforms the unconditional model across all measured metrics: mean squared error (MSE), structural similarity index (SSIM), and signal-to-noise ratio (SNR).

Metric	Unconditional Upsampling	Conditional Upsampling
MSE	0.0053	**0.0015**
SSIM	0.9014	**0.9714**
PSNR (dB)	29.9987	**35.2552**

Table 6.1: Results of Unconditional and Conditional Upsampling on the butterfly dataset.

Figure 6.1 provides a visual comparison of sample outputs from the upsampling models. Additionally, Figure 6.2 shows how the upsampling performance varies with the initial timestep t used in the diffusion process.

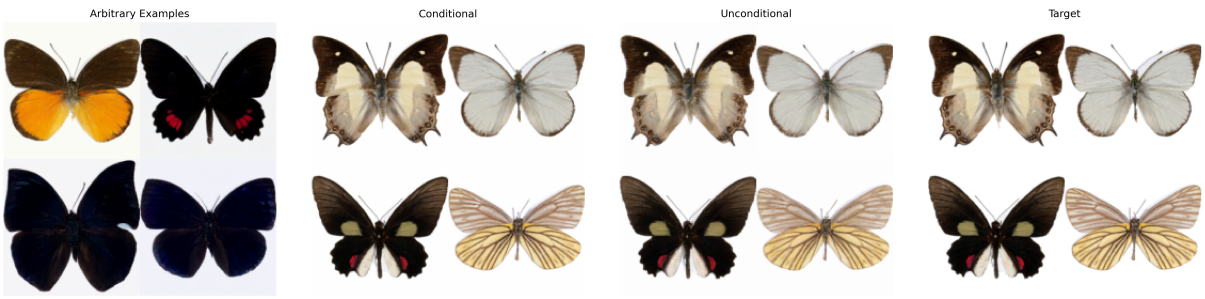

Figure 6.1: Visual comparison of upsampled butterfly images using unconditional and conditional models.

Additional examples of the diffusion model image output and their low-res and and high-res image counterparts, in more clear detail is provided in Appendix 10.1.

6.2 Active Matter Simulation

Table 6.2 summarizes the results for the active matter simulation task. Here, the conditional model yields higher SSIM and SNR values, while the unconditional model achieves a slightly lower FID score.

Sample frames generated by the models are shown in Figure 6.3, along with real validation data for qualitative comparison. To validate the relevance of SSIM and PSNR, we have in appendix

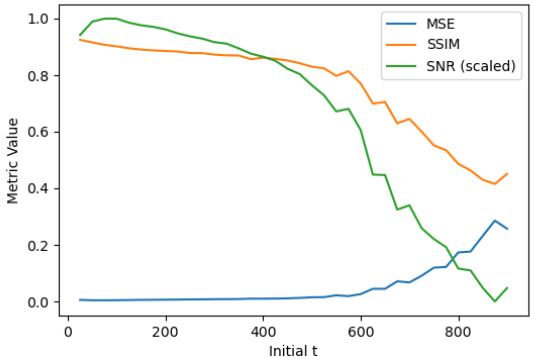

Figure 6.2: Evaluation metrics as a function of initial timestep $t \in [25, 900]$ for the butterfly upsampling task.

Metric	Unconditional Active Matter	Conditional Active Matter
FID	**101.28**	109.99
SSIM	0.981	**0.988**
PSNR (dB)	38.944	**43.878**

Table 6.2: Quantitative evaluation of unconditional and conditional models on active matter simulations.

10.1 figures that show samples of generated images, and their closest matching images in the validation set for both SSIM and PSNR closeness. Showing that whilst the generated data often appear more noisy than the validation set images, we do in some cases find samples that are similar.

6.3 1D Brownian Bridge Simulation

The performance of the diffusion models on the one-dimensional Brownian bridge problem is reported in Table 6.3. The conditional model achieves lower KL divergence values for both distributional targets: the joint endpoint density $\rho(\tau_-, \tau_+)$ and the supremum $\sup|B(t)|$.

Metric	Unconditional Brownian Bridge	Conditional Brownian Bridge
KL - $\rho(\tau_-, \tau_+)$	0.0937	**0.0324**
KL - $\sup\|B(t)\|$	0.270	**0.132**

Table 6.3: KL divergence results for simulated 1D Brownian bridges, evaluating the fit to analytical distributions.

Figure 6.4 shows representative output samples from the diffusion models, while Figure 6.5 compares the empirical and theoretical distributions for the supremum statistic. Showing that the distribution PDF is indeed more similar with the conditional sampling method. We can see from the figure of the samples, that the unconditional model is struggling immensely with creating Brownian Bridges that end in zero, as the training data has had. This is likely due to the problem, when defined by jumps; we have that the model needs to figure out that the sum of all the values should equal zero. This is likely a deceptively difficult problem for a diffusion model to learn, especially, given that the jumps of a Brownian Bridge, does not resemble any dependency structure by the very nature of how Brownian Bridges are constructed.

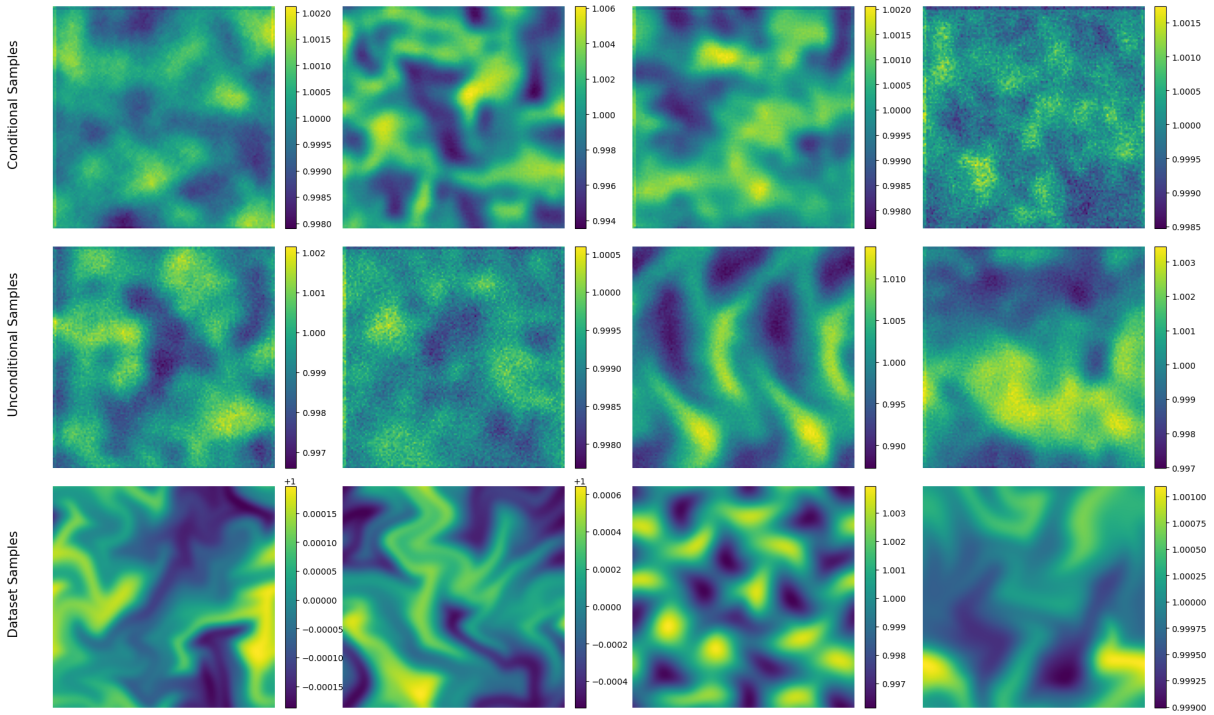

Figure 6.3: Random example frames from active matter simulations.

We have this russian nesting doll of Gaussian noise, where the diffusion model for all t = 1000 is supposed to predict something that looks like gaussian noise ϵ_θ from an input that is also supposed to look like Gaussian noise, in order to calculate an image that is also supposed to resemble a slight scaled Gaussian noise, whilst adding Gaussian noise z. It is quite convoluted to determine whether the problem is in the model, training, or the problem data itself.

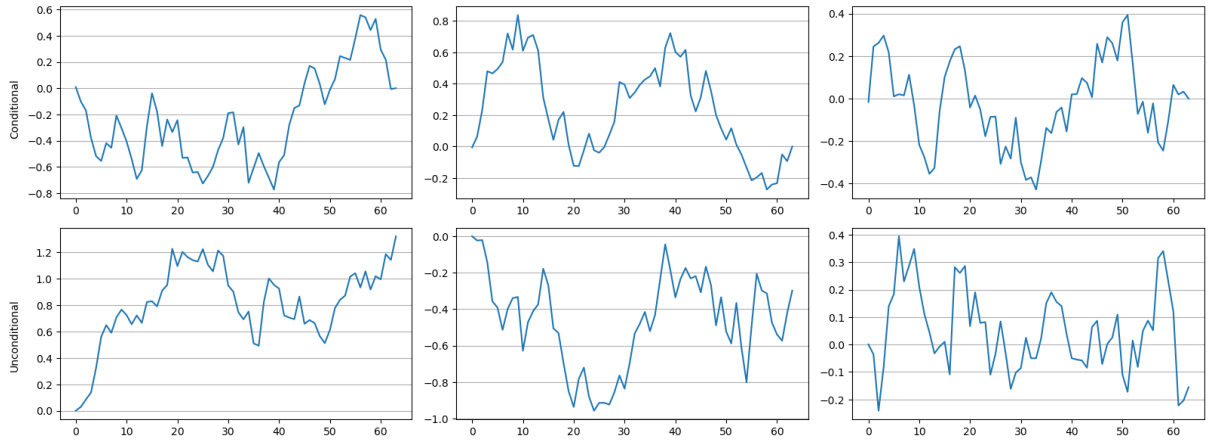

Figure 6.4: Generated trajectories from the trained 1D Brownian bridge diffusion models.

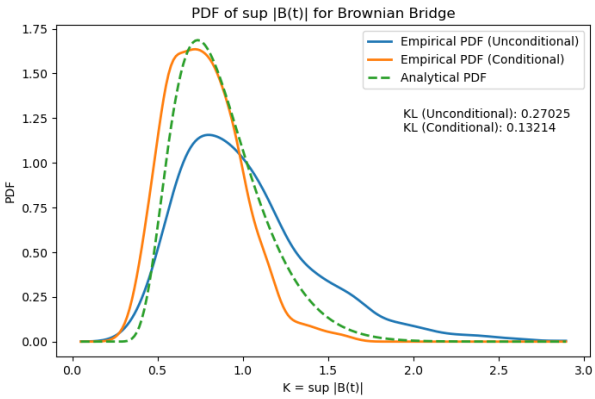

Figure 6.5: Comparison of empirical and theoretical PDFs of $\sup|B(t)|$ for the Brownian bridge task.

7 Discussion

Conclusion

8

Bibliography

9

[Wan+04] Zhou Wang *et al.* „Image quality assessment: From error visibility to structural similarity“. In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612. DOI: 10.1109/TIP.2003.819861.

[He+15] Kaiming He *et al. Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV]. URL: https://arxiv.org/abs/1512.03385.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV]. URL: https://arxiv.org/abs/1505.04597.

[Sal+16] Tim Salimans *et al. Improved Techniques for Training GANs*. 2016. arXiv: 1606.03498 [cs.LG]. URL: https://arxiv.org/abs/1606.03498.

[Vas+17] Ashish Vaswani *et al. Attention Is All You Need*. 2017. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

[Heu+18] Martin Heusel *et al. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium*. 2018. arXiv: 1706.08500 [cs.LG]. URL: https://arxiv.org/abs/1706.08500.

[CF20] Min Jin Chong and David Forsyth. *Effectively Unbiased FID and Inception Score and where to find them*. 2020. arXiv: 1911.07023 [cs.CV]. URL: https://arxiv.org/abs/1911.07023.

[HJA20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. arXiv: 2006.11239 [cs.LG]. URL: https://arxiv.org/abs/2006.11239.

[SME20] Jiaming Song, Chenlin Meng, and Stefano Ermon. „Denoising Diffusion Implicit Models“. In: *CoRR* abs/2010.02502 (2020). arXiv: 2010.02502. URL: https://arxiv.org/abs/2010.02502.

[Bińkowski+21] Mikołaj Bińkowski *et al. Demystifying MMD GANs*. 2021. arXiv: 1801.01401 [stat.ML]. URL: https://arxiv.org/abs/1801.01401.

[ND21] Alex Nichol and Prafulla Dhariwal. *Improved Denoising Diffusion Probabilistic Models*. 2021. arXiv: 2102.09672 [cs.LG]. URL: https://arxiv.org/abs/2102.09672.

[Bet+22] Eyal Betzalel *et al. A Study on the Evaluation of Generative Models*. 2022. arXiv: 2206.10935 [cs.LG]. URL: https://arxiv.org/abs/2206.10935.

[HPJ24] Juno Hwang, Yong-Hyun Park, and Junghyo Jo. „Upsample Guidance: Scale Up Diffusion Models without Training“. In: *arXiv preprint arXiv:2404.01709* (2024). URL: https://arxiv.org/abs/2404.01709.

[Tai+24] Matías Tailanián *et al.* „Diffusion Models Meet Image Counter-Forensics“. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2024, pp. 3925–3935.

[AI25] Polymathic AI. *Active Matter Dataset - The Well*. https://polymathic-ai.org/the_well/datasets/active_matter/. Accessed: 2025-05-05. 2025.

[Wik25] Wikipedia contributors. *Multivariate Normal Distribution — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/wiki/Multivariate_normal_distribution. Accessed: 2025-05-05. 2025.

[Smo] Smoffe. *MSc GitHub Repository*. https://github.com/smoffe1604/MSc/tree/main.

Appendix

10

10.1 Appendix A

10.2 Appendix B

Expanded Diffusion Model and noise schedule details, for both case 2d-super-resolution and Concentration-Conservation:

```
UNet2DModel(
    sample_size=128,
    in_channels=1,
    out_channels=1,
    layers_per_block=2,
    block_out_channels=(128, 128, 256, 256, 512, 512),
    down_block_types=(
        "DownBlock2D", "DownBlock2D", "DownBlock2D",
        "DownBlock2D", "AttnDownBlock2D", "DownBlock2D"
    ),
    up_block_types=(
        "UpBlock2D", "AttnUpBlock2D", "UpBlock2D",
        "UpBlock2D", "UpBlock2D", "UpBlock2D"
    )
)

DDPMScheduler(num_train_timesteps=1000, clip_sample=False)

optimizer = AdamW(model.parameters(), lr=config.learning_rate)
lr_scheduler = get_cosine_schedule_with_warmup(
    optimizer,
    num_warmup_steps=config.lr_warmup_steps,
    num_training_steps=(len(train_dataloader) * config.num_epochs)
)
```

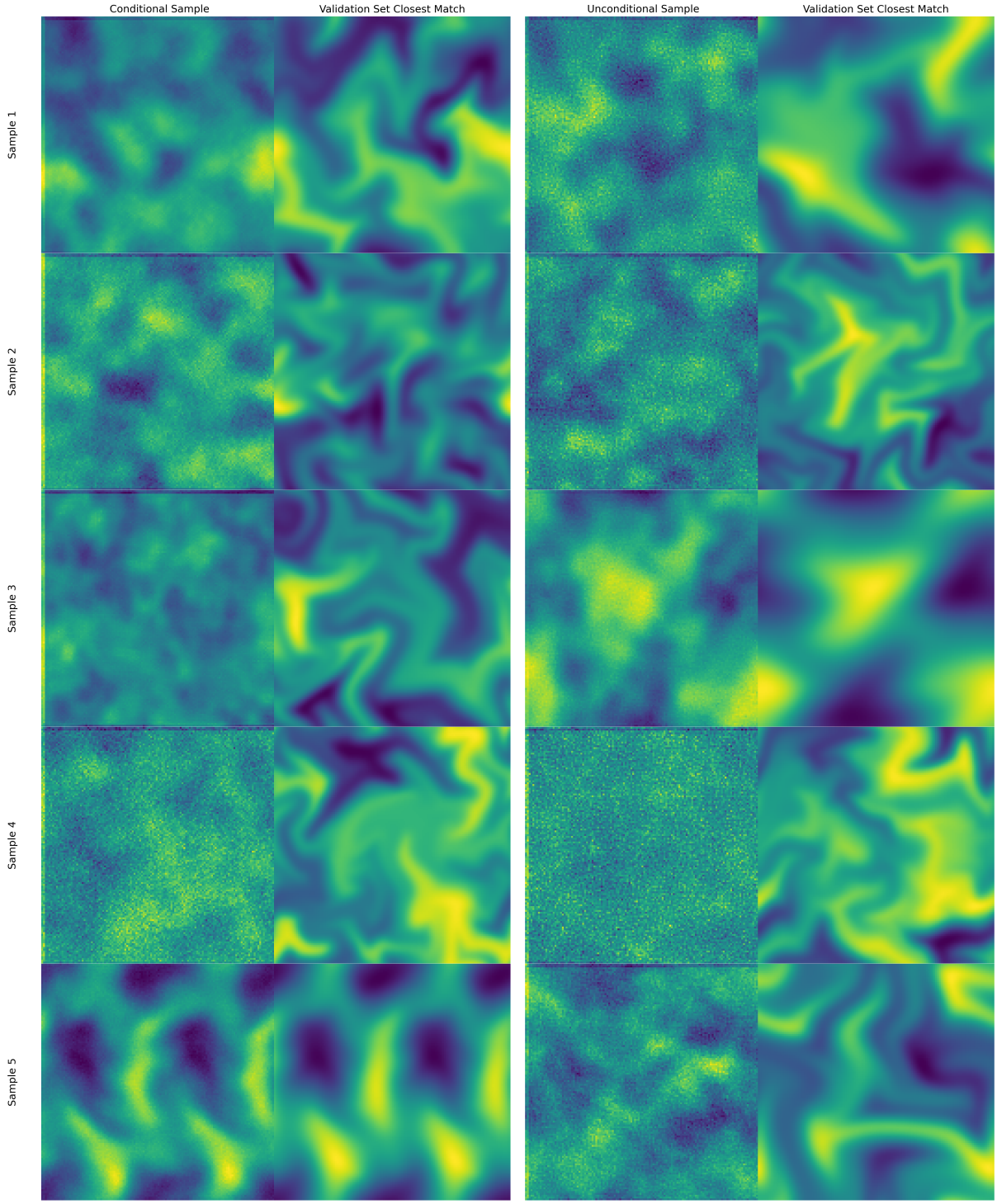

Figure 10.1: Arbitrary sample data, generated from both conditional and unconditional diffusion sampling, compared with their closest matching counterpart in the validation set calculated by Peak-Signal-to-Noise-Ratio (PSNR).

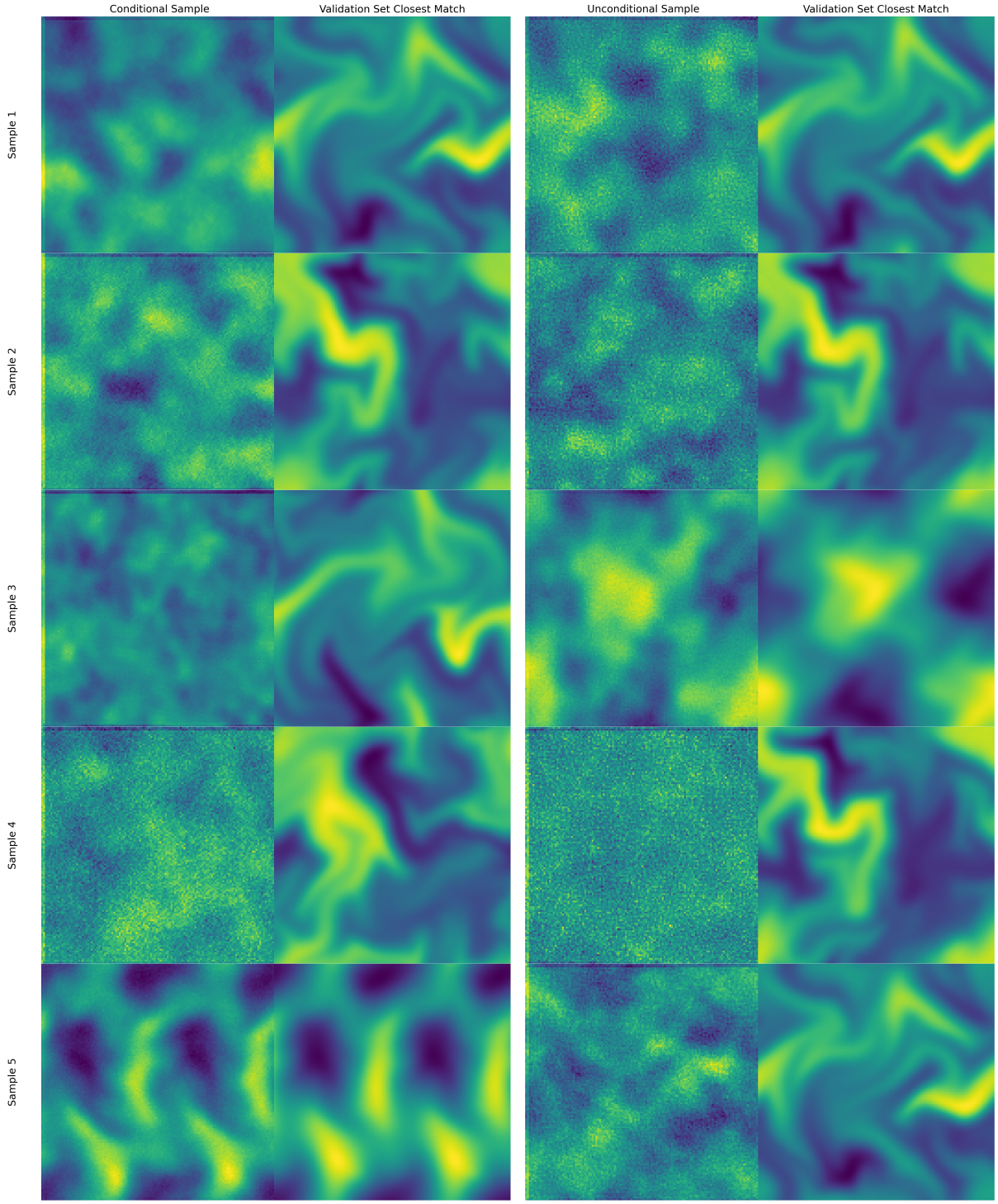

Figure 10.2: Arbitrary sample data, generated from both conditional and unconditional diffusion sampling, compared with their closest matching counterpart in the validation set calculated by Structural similarity index measure (SSIM).

Figure 10.3: Sample images that are produced arbitrarily, to show the capabilities of the base diffusion model.

Figure 10.4: Sample Images of upscaled low-res images from the standard upsampling method with initial t equal to 75.

Figure 10.5: Sample Images of upscaled low-res images from the conditional sampling method.

Figure 10.6: high-res examples of the validation set.

Figure 10.7: Low-res Images of the validation set. Images used as the input to the diffusion sampling process.