Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Enhancing Generative Capabilities of Diffusion Models: Evaluating Particle Guidance and Stochastic Sampling

Semester Thesis

Julian Kleutgens

jkleutgens@ethz.ch

Division of Robotics, Perception and Learning
KTH Stockholm

Supervisors:
Prof. Dr. Josephine Sullivan
Prof. Dr. Ender Konukoglu
M. Sc. Sebastian Gerard

November 30, 2024

Abstract

Diffusion models have emerged as powerful tools in generative modeling, excelling in creating high-quality and diverse outputs. This thesis explores their application to sequential data in a one-scale mapping for segmentation cases, focusing on the primary objectives: enhancing the diversity of generated outputs and evaluating the impact of different sampling processes on diversity and quality. The study leverages Particle Guidance (PG) [1], a method that introduces interactions between generated samples to promote diversity, thus introducing a non-I.I.D sampling process. Additionally, it integrates that into a stochastic sampling framework inspired by the Elucidating the Design Space of Diffusion Models (EDM) [2] paper. Using synthetic datasets such as Moving MNIST and novel variations designed to simulate stochastic movement, we investigate multimode cases where multiple outcomes exist for a given input. The experiments highlight the challenges of maintaining both diversity and quality in generated outputs and explore the impact of factors such as stochastic noise, PG scaling, and kernel selection. Furthermore, we evaluate the model's ability to predict sequential frames while preserving probabilistic priors in movement patterns. Results indicate that PG enhances diversity in multimode cases. However, it also highly negatively influences the quality of the generated data. Hence, there is a trade-off between those two qualities and it's effectiveness is sensitive to scaling and kernel choice.

Contents

Chapter 1

Introduction

Diffusion models, initially successful in generating natural images, are now increasingly used to create semantic segmentation masks conditioned on input images. While various methods address specific challenges in this context, many focus on improving performance by aggregating generated samples. However, the real strength of generative models lies in their ability to produce a diverse range of outputs for a given input, rather than simply refining a single solution.

Research on generating diverse segmentation masks with diffusion models, despite the strong conditioning signals from input images, remains limited. Maintaining diversity is particularly important in scenarios with multiple valid outputs, such as when experts provide different medical opinions or when predicting uncertain future states, like wildfire spread or the movement of an object in a dynamic environment. In these applications, generating a set of diverse, representative outputs that capture the various modes of the underlying distribution is more useful than simply adhering to the conditional data distribution.

The goal is to focus on generating binary segmentation masks instead of multi-class masks. This approach highlights two main challenges: the strong influence of conditioning information and the straightforward denoising in low-noise regions of the diffusion process. These challenges are prominent in the binary case but are likely applicable to the multi-class scenario. Techniques like Analog Bits, which produce multiple binary channels to represent multi-class masks as binary strings at each pixel, could extend these findings to multi-class predictions.

In this thesis, we focus on the limitations and methods to enhance the generative capabilities of diffusion models to produce diverse and representative outputs. Specifically, we delve into the inference and training aspects of diffusion models to achieve greater variety in generated outputs. The segmentation task is represented as binary channel images, and we use a stochastic Moving MNIST dataset, which consists of grayscale images and serves as an equivalent to binary segmentation.

By constructing sequences of movements with the MNIST dataset, the objective is to maximize the variety of images generated during the sampling process.

Using a synthetic dataset like Moving MNIST allows us to define a finite state space of possible outcomes in advance, providing a clear framework for exploration. This approach enables a more controlled and precise evaluation of the diversity achieved during inference.

1.1 Related Work

One suitable use case for this line of research would be making a segmentation map for synthetic dataset of binary segmentation masks, similar to active fire maps. A suitable dataset can be from my supervisor Sebastian Gerard from the paper "WildfireSpreadTS: A dataset of multi-modal time series for wildfire spread prediction" [3]. The paper introduces WildfireSpreadTS, a multi-modal time-series dataset designed to predict the daily spread of wildfires based on satellite observations, fuel, topography, and weather data. It consists of 13,607 images from 607 wildfire events in the U.S. (2018–2021) and offers high spatial (375m) and temporal (24h) resolution, enabling multi-temporal modeling for wildfire prediction. The dataset addresses challenges such as noisy labels, imbalanced data, and high-dimensional inputs, and aims to encourage research on multi-temporal, noise-resilient, or generative methods for wildfire spread prediction.

While segmentation tasks in medical image analysis have been explored in some works, a few are particularly relevant to our study:

A noteworthy example is the paper "Stochastic Segmentation with Conditional Categorical Diffusion Models" [4], which addresses the challenge of generating multiple plausible segmentation maps in scenarios where aleatoric uncertainty is crucial, such as medical diagnostics. This work introduces a Conditional Categorical Diffusion Model (CCDM) that employs categorical diffusion to learn and predict multimodal label distributions conditioned on input images. This study is closely aligned with our work, as it also focuses on using diffusion models to generate diverse segmentation maps, with an emphasis on aleatoric uncertainty. While CCDM handles categorical distributions for multi-class segmentation tasks, our research concentrates on binary segmentation using synthetic datasets like Moving MNIST. Additionally, we explore sequences of previous frames, such as medical diagnostics taken at specific time points, with the goal of predicting future developments in treatment. Despite these differences, both approaches share the overarching goal of leveraging diffusion models to address the challenges of generating multimodal outputs in segmentation tasks.

Another relevant paper, "BerDiff: Conditional Bernoulli Diffusion Model for Medical Image Segmentation," [5] focuses on segmenting medical images to assist in diagnosing and treating illnesses. The paper highlights tasks such as identifying tumor boundaries in brain MRI scans and segmenting lung nodules in CT scans. Instead of relying on Gaussian noise, the authors propose using Bernoulli noise

as the diffusion kernel to enhance the model's capacity for binary segmentation tasks, achieving state-of-the-art results.

Additionally, the paper "Ambiguous Medical Image Segmentation using Diffusion Models" [6] presents findings that align with our thesis. It also leverages the stochastic sampling process of diffusion models to capture the variability inherent in medical image modalities. The study demonstrates performance improvements by using stochastic sampling instead of deterministic approaches, which typically generate a single segmentation output.

1.2 Outline of the Thesis

This brings us to the outline of the thesis. In the next chapter, I will delve into the mathematical foundations and the underlying theory used throughout this work. This chapter will also include an explanation of the adjusted dataset utilized in the experiments and how the techniques from the EDM and PG papers are integrated.

Following this, the performance of the proposed method will be evaluated in both unconditional and conditional generation settings. The conditional generation section will focus on two key aspects: next-frame prediction and estimating segmentation maps over time.

Chapter 2

Background and Methods

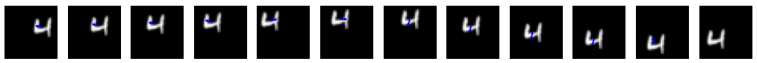
This chapter discusses the methods, implementation, and the technical and theoretical background of the approaches used throughout the thesis. It begins with an explanation of the dataset, followed by a brief overview of diffusion models as introduced previously, focusing on their role in the generation process. Next, the Score-based generative modeling method [7] and key insights from the paper "Elucidating the Design Space of Diffusion-Based Generative Models" (EDM) [2] are examined in detail. Finally, the "Practical Guidance" [1] paper is reviewed, explaining how methods from both papers were implemented. Simple examples of the generation process applied to the dataset are provided throughout.

2.1 Stochastic Moving MNIST

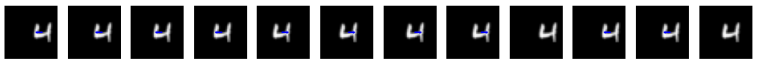
The base of the datasets used in the thesis was drawn from the Moving MNIST which is from the paper: "Stochastic Video Generation with a Learned Prior" [8]. The *Stochastic Moving MNIST (SM-MNIST)* dataset builds on the classic MNIST dataset, which consists of 28×28 grayscale images of handwritten digits ranging from 0 to 9, widely used for digit classification tasks. In SM-MNIST, sequences of frames sized 64×64 feature one or two MNIST digits that move and bounce off the frame edges (walls). Unlike the original Moving MNIST dataset [9], where digits move with constant speed and direction, SM-MNIST introduces variability: each digit follows a consistent trajectory until it hits a wall, at which point it rebounds with a random speed and direction. This blend of predictable motion and random changes upon collision creates segments of deterministic motion mixed with moments of uncertainty. We made several alterations to the dataset for specific use cases. First, the image size was reduced to 32×32, with a single floating digit, also of size 32×32, to simplify the generation process. The following modifications were applied for a different training dataset:

1. **Constant middle**: For generating different digits, each MNIST digit (from a total of 60k digits) was placed at the center of the image using the sizes

(a) Circle: A digit is positioned in the middle of the image and moves to eight random directions back and forth.



(b) Free Directions: Digits from the sequence dataset are moved with a constant velocity. At the border the in the 5th frame the digit gets a new direction.



(c) Horizontal Left and Right: Digits move stochastically left and right, bouncing back at the frame border. At the center, a new random direction is sampled (In the 5th frame).

Figure 2.1: Demonstrations of MNIST digit placements and movements for different dataset variations: (a) Circle, (b) Free Directions, and (c) Horizontal Left and Right. Each variation explores specific aspects of digit generation. The blue arrow indicate in which direction the digit will go next.

specified above. This setup was beneficial for unconditional generation and exploring the diversity of digit appearances.

2. **Free Directions**: This variation used the same sequence dataset as above, but with a constant velocity applied to the digits. This is shown in Figure 2.1b.

3. **Circle**: In this case, a digit was positioned in the middle of the image and then moved to eight random directions back and forth. This was helpful for conditional generation, where the center digit served as the conditional frame, and the model was expected to generate the surrounding conditions. This setup contrasts with the first case, which focused on the types of digits rather than directional movement. This is shown in Figure 2.1a.

4. **Horizontal Left and Right**: The stochastic movement was simplified so that a digit only moves left and right. If it hits the frame border, it bounces back in the opposite direction. When the digit reaches the center of the frame, a stochastic effect is added by randomly sampling a new direction, allowing it to continue either left or right from the center. This is shown in Figure 2.1c.

We aim to explore the effects of particle guidance and stochastic sampling on variation within the generation domain. To maximize evaluation clarity, we focused on simple cases with black-and-white images. This approach not only reduces computational demand but also facilitates a more straightforward evaluation of different cases or modes.

2.2 Diffusion Models

In the field of generative modeling, particularly in computer vision, diffusion models have quickly become effective instruments. These models work in two stages: a reverse denoising phase in which the model reconstructs the original data from the noisy input, and a forward diffusion phase in which data is progressively disturbed by the addition of noise. As strong alternatives for other generative models such as GANs (Generative Adversarial Networks) [10] and VAEs (Variational Autoencoders) [11], diffusion models are excellent at producing a wide range of high-quality samples. However, because each sample must be generated through a number of repeated processes, one of the main issues with diffusion models is still their high computational needs. In exploring diffusion-based generative models, an area of interest is the exposition of their design space. Key designs include Denoising Diffusion Probabilistic Models (DDPMs) [12], which train a sequence of models to gradually reverse the process of adding noise to data. It uses knowledge about the structure of each step in this reverse process to simplify training. For continuous state spaces, the DDPM training objective

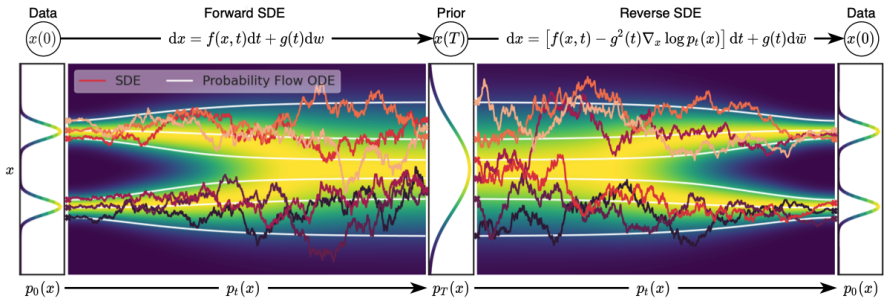


Figure 2.2: Overview of score-based generative modeling through SDEs from the paper. The figure illustrates the forward and reverse stochastic differential equations (SDEs) used in this method. In the forward process, the data is mapped to a noise distribution. Here, the two peaks at $p_0(x)$ in the distribution could represent two distinct classes of images, which are gradually transformed into a noise distribution.

implicitly computes scores at each noise scale and, therefore refers to score-based generative models.

2.3 Score-Based Generative Modeling through Stochastic Differential Equations

The paper [7] develops new sampling methods and expands the capabilities of score-based generative models. The authors introduce a flexible framework that builds on previous approaches using stochastic differential equations (SDEs). Unlike DDPMs, which add noise in a few fixed steps in a discrete manner, this framework employs a continuous series of noise distributions that evolve over time, following a diffusion process. This process gradually transforms data into random noise through a fixed SDE, independent of the data and without any trainable parameters. By inverting this process, random noise can be transformed back into structured data, enabling smooth sample generation. An overview of their framework is shown in Figure 2.2. The goal is to create a diffusion process $\{x(t)\}_{i=0}^{T}$ for t in the continuous range $t \in [0, T]$. Here, $x(0) \sim p_0$ represents the data distribution, for which we have independent samples in our dataset, and $x(T) \sim p_T$ is a chosen prior noise distribution that we can easily sample from. Typically, p_T is an unstructured distribution that contains no information of p_0.

This diffusion process follows an Ito SDE (Stochastic Differential Equation):

$$dx = f(x,t)\,dt + g(t)\,dw \tag{2.1}$$

Where w is a standard Wiener process (or Brownian motion). In this equation:

- $f(x,t) : \mathbb{R}^d \to \mathbb{R}^d$ is a vector-valued function called the *drift coefficient*, which determines the direction and rate of change of $x(t)$, where d is the dimensionality of the dataset. This is what one would have in the "probability flow" ordinary differential equation and covers the deterministic properties.
- $g(t) : \mathbb{R} \to \mathbb{R}$ is the *diffusion coefficient*, a scalar function that controls the amount of randomness added to $x(t)$ at each time point from the Brownian motion dw

These coefficients are selected differently for the variance preserving (VP) and variance exploding (VE) formulations. Inverting the forward process from samples of $x(T) \sim p_T$ to samples $x(0) \sim p_0$ requires solving a corresponding reverse-time SDE, which essentially undoes the noise perturbations by using the learned score function:

$$dx = \left[f(\mathbf{x},t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right] dt + g(t) d\bar{w}, \tag{2.2}$$

In this setup, $\bar{w}$ represents a standard Wiener process as time moves in reverse, from T down to 0, with dt being a small negative time increment (infinitesimal). If we have the score function, $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$, which provides the gradient of the log-probability for each distribution at time t, then we can derive the reverse of the diffusion process. The paper points out that one can train a time-dependent score-based model $\mathbf{s}_\theta(\mathbf{x}, t)$ via a continuous generalization to estimate $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$. This model is normally defined with a network similar to the U-Net [13]. It is effective because of its symmetric encoder-decoder structure with skip connections, allowing for detailed information from early layers to be retained and combined with deeper, coarser features. These skip connections help the model maintain fine details and spatial structure throughout the network. The following section explains the EDM [2] which builds up on the score-based generative models.

2.4 Elucidating the Design Space of Diffusion-Based Generative Models (EDM)

In the section before we only talked about the time steps t where the timesteps are the diffusion to the noise for higher T. Now, to be aligned with the paper's definition, the common framework for expressing diffusion models is expressed with sigma and t. Let $p_{\text{data}}(x)$ represent the data distribution with standard deviation σ_{data}. Define the mollified distributions $p(x;\sigma)$ by adding i.i.d. Gaussian noise of standard deviation σ. For $\sigma_{\max} \gg \sigma_{\text{data}}$, $p(x;\sigma_{\max})$ resembles pure Gaussian noise. Diffusion models start with $x_0 \sim \mathcal{N}(0, \sigma_{\max}^2 \mathbf{I})$ and progressively denoise to obtain x_i at noise levels $\sigma_0 = \sigma_{\max} > \sigma_1 > \cdots > \sigma_N = 0$, where $x_i \sim p(x_i;\sigma_i)$. The endpoint x_N aligns with the data distribution. Note that,

the higher indexes represent less noise. In comparison to the Equation 2.2 they formulated the ODE differently. First, one chooses a schedule $\sigma(t)$ that defines the desired noise level at time t (e.g. $\sigma(t) \sim \sqrt{t}$). The schedule makes a big difference in the generation process. Note that, this is without the Wiener process term $g(t)d\bar{w}$, so this is the deterministic case.

$$dx = -\dot{\sigma}(t)\,\sigma(t)\,\nabla_x \log p(x;\sigma(t))\,dt, \tag{2.3}$$

They demonstrate this in Appendix B.1 and B.2 of the paper. In Equation 39, they arrive at the exact solution. The key adjustment to achieve this is expressing the flow ODE from Equation 2.2 in terms of $p(x,\sigma)$ rather than $p_t(x)$.

$$p(x;\sigma) = p_{\text{data}} * \mathcal{N}(0,\sigma(t)^2\mathbf{I}) \quad \text{and} \quad p_t(x) = s(t)^{-d}\, p\left(\frac{x}{s(t)};\sigma(t)\right). \tag{2.4}$$

The $*$ symbol represents the convolution of probability density functions, with an additional scale schedule $s(t)$. The convolution represents a smoothed version of the original data distribution, created by adding independent Gaussian noise to each sample. $\nabla_x \log p(x;\sigma(t))$ is the score function, a vector field that points toward areas with higher data density at a given noise level. Taking a small forward step along this ODE moves the sample away from the data distribution, with the rate of movement depending on the change in noise level. Conversely, a backward step guides the sample toward the data distribution. Next they introduce a denoiser function $D(x;\sigma)$ that minimizes the expected L2 between a noise image given σ and the true image.

$$\mathbb{E}_{y\sim p_{\text{data}}}\mathbb{E}_{n\sim\mathcal{N}(0,\sigma^2\mathbf{I})}\|D(y+n;\sigma)-y\|_2^2, \text{ then } \nabla_x \log p(x;\sigma) = \frac{D(x;\sigma)-x}{\sigma^2} \tag{2.5}$$

$$D_\theta(x;\sigma) = c_{\text{skip}}(\sigma)x + c_{\text{out}}(\sigma)F_\theta\left(c_{\text{in}}(\sigma)x; c_{\text{noise}}(\sigma)\right) \tag{2.6}$$

In Equation 4.1 is the definition of the denoiser while F_θ represents a neural network (like the U-Net). The shape of the solution paths for the ODE depends on the functions $\sigma(t)$ and $s(t)$. By carefully choosing these functions, one can minimize truncation errors, which are proportional to the curvature of $\frac{dx}{dt}$. In the paper, they recommend setting $\sigma(t) = t$ and $s(t) = 1$, which is the same choice used in DDIM [14]. With these settings, the ODE solution matches the form given in Equation 2.3. The general solution of the ODE includes additional terms, which are detailed in Appendix B.2, Equation (38). These terms are set to zero because $\dot{s}(t) = 0$.

One of the contributions of the paper is to use Heun's 2nd order method instead of the (improved Euler). Using Heun's 2nd order method allows us to make each step more accurate (reducing truncation error) without needing a ton of steps (lower neural function evaluations). So, Heun's method finds a balance between accuracy and efficiency.

This was the theoretical concept behind training and sampling in a deterministic manner. However, as mentioned in the previous section, we are particularly interested in the stochastic sampler, specifically the $g(t)\,d\bar{w}$ term in Equation 2.2. This term introduces fresh noise into the image at each denoising step. The EDM paper incorporates this term into their SDE in Appendix B.4.

$$d\mathbf{x}_{\pm} = \left(\frac{1}{2}g(t)^2 - \dot{\sigma}(t)\sigma(t)\right)\nabla_{\mathbf{x}}\log p(\mathbf{x};\sigma(t))\,dt + g(t)\,d\omega_t \tag{2.7}$$

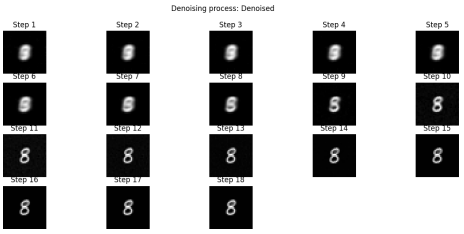
$$= \underbrace{-\dot{\sigma}(t)\sigma(t)\nabla_{\mathbf{x}}\log p(\mathbf{x};\sigma(t))\,dt}_{\text{probability flow ODE (Equation 2.3)}} \pm \underbrace{\underbrace{\beta(t)\sigma(t)^2\nabla_{\mathbf{x}}\log p(\mathbf{x};\sigma(t))\,dt}_{\text{deterministic noise decay}} + \underbrace{\sqrt{2\beta(t)\sigma(t)}\,d\omega_t}_{\text{noise injection}}}_{\text{Langevin diffusion SDE}} \tag{2.8}$$

They set $g(t) = \sqrt{2\beta(t)\sigma(t)}$. This term allows for both forward $d\mathbf{x}_+$ and backward $d\mathbf{x}_-$ movement in time. In the Langevin term, we observe a deterministic score-based denoising component alongside a stochastic noise injection term. Langevin diffusion guides the sample toward the target marginal distribution at a specific time, actively correcting any errors from earlier sampling steps. This leads to the pseudocode, which combines a second-order deterministic approach with the addition and removal of noise.

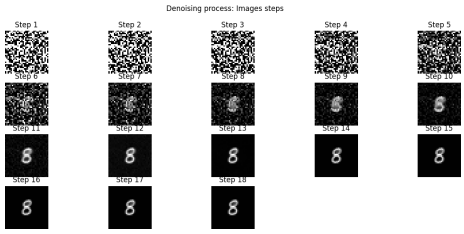
Algorithm 1 EDM stochastic sampler with $\sigma(t) = t$ and $s(t) = 1$

procedure StochasticSampler($D_\theta(\mathbf{x};\sigma), \{t_i\}_{i=0,\dots,N}, \{\gamma_i\}_{i=0,\dots,N-1}, S_{\text{noise}}$)
  sample $\mathbf{x}_0 \sim \mathcal{N}(0, t_0^2\mathbf{I})$
  for $i \in \{0,\dots,N-1\}$ **do**
    sample $\epsilon_i \sim \mathcal{N}(0, S_{\text{noise}}^2\mathbf{I})$
    $\hat{t}_i \leftarrow t_i + \gamma_i t_i$ ▷ Select temporarily increased noise level $\hat{t}_i$
    $\hat{\mathbf{x}}_i \leftarrow \mathbf{x}_i + \sqrt{\hat{t}_i^2 - t_i^2}\,\epsilon_i$ ▷ Add new noise to move from t_i to $\hat{t}_i$
    $\mathbf{d}_i \leftarrow (\hat{\mathbf{x}}_i - D_\theta(\hat{\mathbf{x}}_i;\hat{t}_i))/\hat{t}_i$ ▷ Evaluate $d\mathbf{x}/dt$ at $\hat{t}_i$
    $\mathbf{x}_{i+1} \leftarrow \hat{\mathbf{x}}_i + (t_{i+1} - \hat{t}_i)\,\mathbf{d}_i$ ▷ Take Euler step from $\hat{t}_i$ to t_{i+1}
    if $t_{i+1} \neq 0$ **then**
      $\mathbf{d}'_i \leftarrow (\mathbf{x}_{i+1} - D_\theta(\mathbf{x}_{i+1};t_{i+1}))/t_{i+1}$ ▷ Apply 2nd order correction
      $\mathbf{x}_{i+1} \leftarrow \hat{\mathbf{x}}_i + (t_{i+1} - \hat{t}_i)\left(\frac{1}{2}\mathbf{d}_i + \frac{1}{2}\mathbf{d}'_i\right)$
    end if
  end for
  return $\mathbf{x}_N$
end procedure

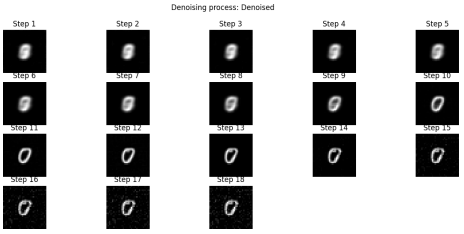
The added noise to the sampling process is factored through $\gamma_i \geq 0$. Note, however, that when $\gamma_i = 0$, the sampling reverts to a deterministic process. It is important to mention that in the pseudocode notation, $t_i(= \sigma(t_i))$ represents the noise level, where t_N corresponds to no noise and t_0 represents the initial noise.



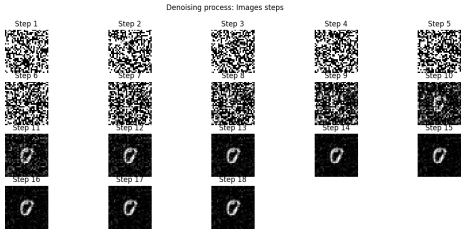
(a) Deterministic: Output of the denoiser, $D_\theta(\hat{\mathbf{x}}_i)$.



(b) Deterministic: Denoising steps over time, $\mathbf{x}_n$, $n \in \{1, \ldots, 18\}$.



(c) Stochastic: Output of the denoiser, $D_\theta(\hat{\mathbf{x}}_i)$.



(d) Stochastic: Denoising steps over time, $\mathbf{x}_n$, $n \in \{1, \ldots, 18\}$.

Figure 2.3: Comparison of the **deterministic** and **stochastic** denoising processes. The deterministic process progresses steadily, while the stochastic process retains noise for a longer duration.

In the implementation and throughout this report, $\sigma(t_0) = 80$ and $\sigma(t_N) = 0.002$. The sampler based on this algorithm was used for all experiments conducted in this project.

An important aspect to mention is the preconditioning discussed in Section 5 of the paper, represented by $c_{\text{skip}}(\sigma)$, $c_{\text{out}}(\sigma)$, $c_{\text{in}}(\sigma)$, and $c_{\text{noise}}(\sigma)$ in Equation 4.1. The main idea is to simplify the task for the U-Net by feeding inputs with unit standard deviation via $c_{\text{in}}(\sigma)$. This means that the noise in the images has a mean of zero and a standard deviation of one. To reverse this scaling, $c_{\text{out}}(\sigma)$ adjusts the output magnitudes. Additionally, $c_{\text{skip}}(\sigma)$ modulates the skip connection, and $c_{\text{noise}}(\sigma)$ maps the noise level σ to a conditioning input for F_θ. The exact definitions of these coefficients are provided in the appendix of the EDM paper [2]. Figure 6.1 gives an overview of the model's architecture.

Additionally, it is important to note that this architecture supports both conditional and unconditional generation. For unconditional generation, the neural network serves as a direct input-output denoiser, while for conditional generation, the network takes both an input and a class label to produce an output.

2.4.1 Example Inference on the MNIST Dataset

To illustrate the behavior of these parameters during inference, we present an example where the model was trained on MNIST images. In this case, the model was trained on the static middle frame across 60,000 different digits. The original CNN model contains 43 million trainable parameters, having been initially trained on color images from datasets like CIFAR-10, ImageNet, AFHQv2, and FFHQ. This parameter count is excessive for our case with grayscale images. Therefore, we scaled down the trainable parameters to 6 million, which proved sufficient for our needs. The computational costs were manageable using two A100 GPUs (40 GB). Reducing the number of U-Net blocks and limiting the number of channels in the deeper layers of the U-Net significantly decreases the parameter count.

In Figure 2.3b, one sees the resulting generation process for a specific seed. The image of the digit '8' is generated consistently when using the same seed for the initial noisy image. In Figure 6.2, the directional vectors, $\mathbf{d}'_i$ and $\mathbf{d}_i$, are plotted.

In Figure 2.3d, we show the resulting denoising process using the full stochastic sampler from Algorithm 1. Before, γ_i is set to zero for deterministic sampling. In the Python code implementation, however, γ_i is defined as $\gamma_i = \frac{S_{\text{churn}}}{N}$, making it constant across the denoising steps but decreasing with a higher number of defined steps. Throughout the project, S_{noise} is set to one. The parameter S_{churn} is crucial for tuning; in Figure 2.3d, it is set to 0.7.

It is important to note that both the deterministic and stochastic processes start from the same initial noisy image. However, in the stochastic process, a zero is generated instead, and this outcome can vary with each new run. In Figure 2.3c, the denoising initially seems to form an eight but shifts to a zero as the process continues. When comparing the images, one can observe that the stochastic process requires nearly twice as many steps to achieve the same denoising level as the deterministic process. This behavior aligns with the theoretical understanding, which suggests that the type of image, in our case digit, is determined in high-noise regions.

2.5 Partial Guidance

In this section, we explain the paper "Particle Guidance: Non-I.I.D. Diverse Sampling with Diffusion Models" [1] and describe how we integrated this technique into the stochastic sampler from the EDM paper.

Following the foundational approach from Classifier Guidance (CG), where a guiding term is incorporated into the reverse diffusion process, PG aims to enhance the diversity of generated samples without requiring retraining of the

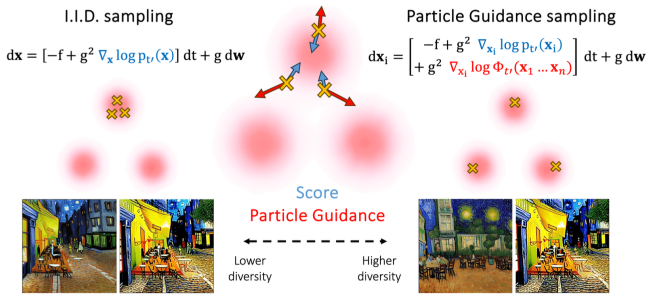


Figure 2.4: This is from the original paper, where the idea of the Particle Guidance is illustrated. In the figure it shows the distribution in pink and the samples as yellow crosses. The two different arrows are the score (in blue) and the guidance from joint-potential (red)

score function. Particle Guidance (PG) extends the setup of the stochastic differential equation (SDE) by sampling not individual data points but rather a set of particles $x_1, \ldots, x_n$. Here, the reverse diffusion SDE is modified to include a joint potential function $\log \Phi_t$ that is permutation-invariant across all particles. This joint potential introduces interactions among particles during the sampling process, encouraging diversity within the particle set and ensuring that they are not independently and identically distributed (non-I.I.D.). As a result, the particles interact and diversify as they are generated, while still benefiting from the underlying diffusion model framework. This concept is illustrated in Figure 2.4, where three particles, represented as yellow points, are shown, so $n = 3$ in the equation. It demonstrates that the particles can now explore new data regions. The red meshes correspond to different modes and are represented differently across cases. For example, in the MNIST case, we have ten distinct meshes for each of the ten digits.

With this setup, the author introduces a new SDE:

$$d\mathbf{x}_i = \left[-f(\mathbf{x}_i, t') + g^2(t') \left(\nabla_{\mathbf{x}_i} \log p_{t'}(\mathbf{x}_i) + \nabla_{\mathbf{x}_i} \log \Phi_{t'}(\mathbf{x}_1, \ldots, \mathbf{x}_n)\right)\right] dt + g(t') d\mathbf{w}. \tag{2.9}$$

$$\log \Phi_t(\mathbf{x}_1, \ldots, \mathbf{x}_n) = -\frac{\alpha_t}{2} \sum_{i,j} k_t(\mathbf{x}_i, \mathbf{x}_j) \tag{2.10}$$

$$d\mathbf{x}_i = \left[-f(\mathbf{x}_i, t') + g^2(t') \left(\nabla_{\mathbf{x}_i} \log p_{t'}(\mathbf{x}_i) - \alpha_{t'} \nabla_{\mathbf{x}_i} \sum_{j=1}^{n} k_{t'}(\mathbf{x}_i, \mathbf{x}_j)\right)\right] dt + g(t') d\mathbf{w} \tag{2.11}$$

Where $k_t(\mathbf{x}_i, \mathbf{x}_j)$ is a kernel that is free to choose. In the paper they use the Radial basis function kernel $k_t(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{h_t}\right)$.

2.5.1 Integration with EDM Sampler

We integrate this Particle Guidance technique into the stochastic sampler derived from the EDM paper, modifying the standard sampling procedure to incorporate PG. The resulting pseudocode for that is in Algortithm 2.

Algorithm 2 EDM stochastic sampler with Particle Guidance

1: **procedure** SAMPLER($D_\theta(\mathbf{x};\sigma), \{t_i\}_{i=0,\dots,N}, \{\gamma_i\}_{i=0,\dots,N-1}, S_{\text{noise}}, k_{\text{PG}}$, B)
2: sample $\mathbf{x}_0^b \sim \mathcal{N}(0, t_0^2\mathbf{I})$ with $b \in \{1,\dots,B\}$ such that $\mathbf{X}_0 = [\mathbf{x}_0^1, \cdots, \mathbf{x}_0^B]$
3: **for** $i \in \{0,\dots,N-1\}$ **do**
4: sample $\epsilon_i \sim \mathcal{N}(0, S_{\text{noise}}^2\mathbf{I})$
5: $\hat{t}_i \leftarrow t_i + \gamma_i t_i$ ▷ Select temporarily increased noise level $\hat{t}_i$
6: $\hat{\mathbf{X}}_i \leftarrow \mathbf{X}_i + \sqrt{\hat{t}_i^2 - t_i^2}\,\epsilon_i$ ▷ Add new noise to move from t_i to $\hat{t}_i$
7: $\hat{\mathbf{X}}_{D_i} \leftarrow D_\theta(\hat{\mathbf{X}}_i; \hat{t}_i))$ ▷ Get the denoised images
8: $\mathbf{d}_i \leftarrow (\hat{\mathbf{X}}_i - \hat{\mathbf{X}}_{D_i})/\hat{t}_i$ ▷ Evaluate $d\mathbf{X}/dt$ at $\hat{t}_i$
9: $\mathbf{d}_{\text{PG}}^b \leftarrow \nabla_{\mathbf{x}_b} \sum_{j=1}^{B} k_{t'}(\mathbf{x}_b, \mathbf{x}_j)$ ▷ PG is computed for all x in batch
10: $\mathbf{d}_i \leftarrow \mathbf{d}_i - k_{\text{PG}}(t_i) * \mathbf{d}_{\text{PG}}$ ▷ Update the $d\mathbf{X}/dt$ with PG
11: $\mathbf{X}_{i+1} \leftarrow \hat{\mathbf{X}}_i + (t_{i+1} - \hat{t}_i)\,\mathbf{d}_i$ ▷ Take Euler and PG step
12: **if** $t_{i+1} \neq 0$ **then**
13: $\mathbf{d}'_i \leftarrow (\mathbf{X}_{i+1} - D_\theta(\mathbf{X}_{i+1}; t_{i+1}))/t_{i+1}$ ▷ Apply 2nd order correction
14: $\mathbf{X}_{i+1} \leftarrow \hat{\mathbf{X}}_i + (t_{i+1} - \hat{t}_i)\left(\frac{1}{2}\mathbf{d}_i + \frac{1}{2}\mathbf{d}'_i\right)$
15: **end if**
16: **end for**
17: **return** $\mathbf{X}_N$
18: **end procedure**

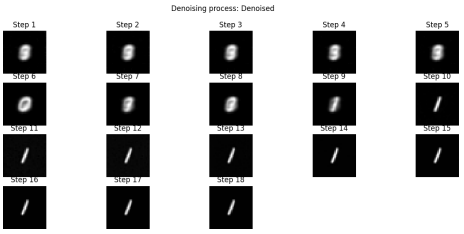
The execution of line 9 in the algorithm highly depends on the definition of the kernel $k_t(\mathbf{x}_i, \mathbf{x}_j)$ and the distance $d(\mathbf{x}_i, \mathbf{x}_j)$. For the start we radial basis function kernel this would result in the following implementation:

$$\nabla_{\mathbf{x}_b} \sum_{j=1}^{B} k_t(\mathbf{x}_b, \mathbf{x}_j) = \nabla_{\mathbf{x}_b} \sum_{j=1}^{B} \exp\left(-\frac{d(\mathbf{x}_b, \mathbf{x}_j)}{h_t}\right) \tag{2.12}$$

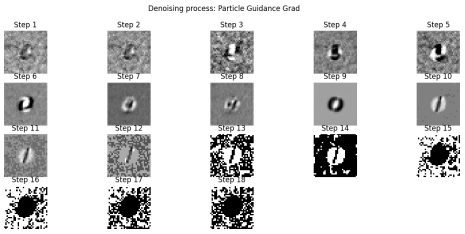
$$= \nabla_{\mathbf{x}_b} \sum_{j=1}^{B} \exp\left(-\frac{\|\mathbf{x}_b - \mathbf{x}_j\|^2}{h_t}\right) \tag{2.13}$$

$$= -\frac{2}{h_t} \sum_{j=1}^{B} (\mathbf{x}_b - \mathbf{x}_j) k_t(\mathbf{x}_b, \mathbf{x}_j) \tag{2.14}$$

Additionally, it is important to note that we scaled k_{PG} with t_i, giving particle guidance a stronger influence in high-noise regions. Such that $k_{\text{PG}}(t_i) = k_{\text{PG}} \cdot t_i$ The rationale behind this is that, in high-noise regions, the type of image (in our case, the digit) is determined. Particle guidance should have a greater effect here,



(a) Denoised output with particle guidance applied, showcasing the effect of the guidance on image generation.



(b) Gradient visualization of the particle guidance, representing the directional influence in the generation process.

Figure 2.5: Illustration of particle guidance: denoised output (left) and corresponding guidance gradient (right).

as we aim to push particles away from each other to explore the generative distribution and capture different modes. In contrast, in low-noise regions, we want the model to focus on denoising and achieving high image quality, as diversity is no longer needed at this stage.

In Figure 2.5, the denoising steps of the image are shown alongside the particle guidance gradient. Note that this uses the same seed as the previous figure; however, this time the model generates a one instead of an eight. Particle guidance is also deterministic when using the same latent noise images with the same batch size for each run.

Chapter 3

Unconditional Generation

In this chapter, I will build on the model presented in the background section. Our aim here is to test the particle guidance and EDM sampler on black-and-white images. Specifically, we seek to generate the different digits from the MNIST dataset. The goal is to evaluate how effectively we can reproduce the results from the paper. While the original paper uses color images, where determining if all modes are discovered is challenging, the MNIST dataset provides a clearer framework. Here, the ten digits represent distinct modes that the model should explore.

The goal is to reproduce the results shown in the appendix of the paper, specifically Figure 6.3, where a heatmap illustrates the number of modes discovered in each run with different configurations. For this demonstration, the authors simplified the problem to a 2D vector space, where the desired modes are represented by ten different Gaussian mixtures. For this theoretical experiment, they developed an inference schedule similar to the denoising process, using three vectors: the score (pointing in the direction of the nearest Gaussian), the particle guidance gradient, and a stochastic random vector. This setup is equivalent to the three gradients in our sampling algorithm 2, except that we also incorporate a second-order method.

In the results, it is evident that a higher particle guidance factor leads to more modes being discovered, creating a heatmap with a smooth gradient of brightness. In Figure 6.3, it is worth noting that the stochastic gradient negatively impacts the number of discovered modes. However, in the context of diffusion models, many papers have shown that diversity improves with added noise, as it serves as a correction term, as referenced in score-based methods [7].

To begin, we trained the diffusion model as explained in the background section, using a schedule of $\sigma(t) = t$. The training data consisted of MNIST digits centered in the middle of 32×32 images. We trained for 20,000 iterations with a batch size of 512 in an unconditional setting.

For digit detection, a Residual Network (ResNet) [15] proved to be a strong choice, specifically the ResNet16 variant. The ResNet is a convolutional multi-

layer backbone commonly used in classification tasks due to its high accuracy. For MNIST, an initial convolution layer was added to accommodate an input channel size of three, as required by the pretrained ResNet. In the final layer, a fully connected layer with ten nodes was added, one for each digit. For training, we used the dataset with 16x16 digits centered in a 32x32 image, utilizing all 60,000 digits from the original MNIST dataset over five epochs. Validation on 1,000 images yielded an accuracy of 99.6%, which was sufficient for accurately detecting digits in the frames generated by the diffusion model.

Since the model tended to overfit and the background of the original MNIST images is set to zero (while the generated images from the diffusion model only approximate zero), misclassifications occurred. To address this, we applied the Otsu method [16] to mask the background from the generated digit effectively.

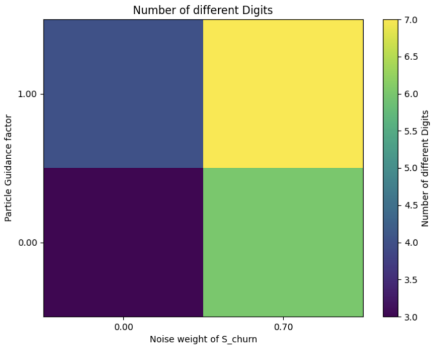
For the generation, we used the sampler described in the previous chapter with configurations of 22 denoising steps ($N = 22$), $S_{\text{churn}} \in \{0, 0.7\}$ (in Algorithm 1, $\gamma_i = \frac{S_{\text{churn}}}{N}$) for adding noise, and particle guidance $k_{\text{pg}} \in \{0, 1\}$. In Figure 3.1 (a), the number of distinct digits generated is shown for four different cases. The specific digits generated are visualized in (b) for the deterministic case and in (c) for the cases where particle guidance and stochastic sampling are applied.

It is important to note that the same seed for the initial noisy image is used across all cases. Therefore, the deterministic cases in the left column of the figure produce identical outcomes for every run. Comparing the particle guidance case (top-left square) to the deterministic one (bottom-left square), we observe that particle guidance generates one additional unique digit, transforming one of the four generated eights into a one. The corresponding UMAP [17] visualization is shown in Appendix Figure 6.4.

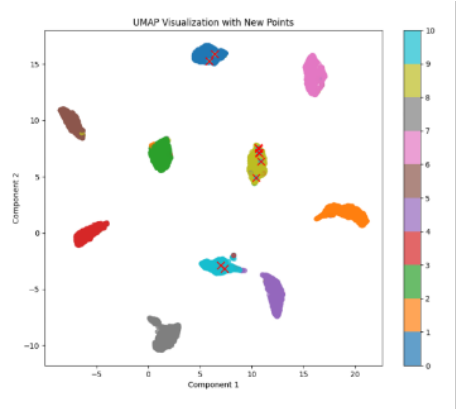
In general, the rows compare the presence or absence of particle guidance, while the columns indicate whether stochastic sampling is applied. For stochastic sampling, a significant improvement is observed, with two additional digits generated. Comparing the UMAPs, the particles are more evenly distributed across the data distribution, now covering 7 out of the 10 digits.

For this visualization, we use the unsupervised technique Uniform Manifold Approximation and Projection (UMAP) [17]. UMAP generates a low-dimensional embedding of high-dimensional data through two main steps: first, it constructs a weighted k-nearest neighbor graph that captures local relationships between data points using a smooth nearest-neighbor distance metric; second, it optimizes the layout of this graph in the low-dimensional space by minimizing the cross-entropy between the high-dimensional and low-dimensional representations. In our case, the high-dimensional data corresponds to the feature maps extracted from a trained ResNet, which is used to detect the digit.

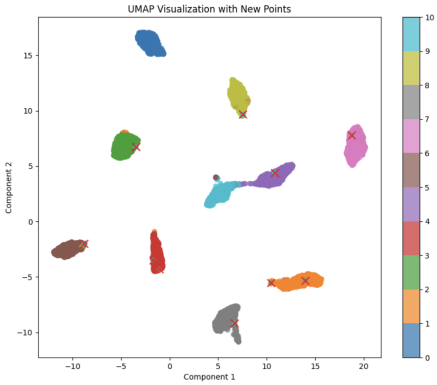
To further explore the effects of particle guidance and stochastic sampling,



(a) Number of modes generated with four configurations.



(b) Deterministic config with particle guidance 0, stochastic sampling at 0 (bottom left).



(c) Particle guidance stage 1, stochastic sampling 0.7 (top right).

Figure 3.1: Comparison of UMAP visualizations: (a) heatmap overview showing mode counts, (b) particle guidance stage 0, and (c) particle guidance stage 1. The red crosses represent the resulting ten particles after the generation process.

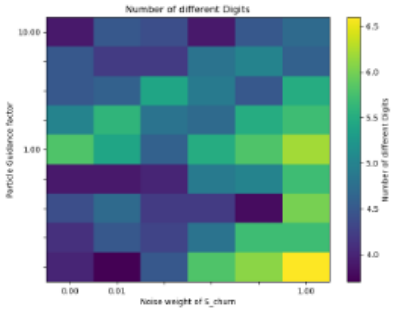
we conducted experiments with the following configurations:

$$k_{\text{PG}} = 10^{c_{\text{PG}}} \quad \text{with} \quad c_{\text{PG}} \in \{-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1\}, \tag{3.1}$$

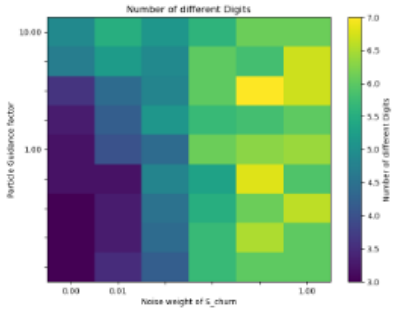
$$S_{\text{churn}} = \{0, 10^{c_{\text{churn}}}\} \quad \text{with} \quad c_{\text{churn}} \in \{-2, -1.5, -1, -0.5, 0\}. \tag{3.2}$$

This resulted in a total of $6 \times 9 = 54$ runs. For each run, 10 batches were sampled, with each batch containing 10 images. Therefore, 100 images were sampled for each of the 54 runs. The ResNet was used to identify the digits generated in each batch, and the mean number of unique digits was calculated for each configuration.

In Figure 3.2 (a), the results from the original implementation using the kernel from Equation 2.11 for particle guidance are shown. Unfortunately, the heatmap



(a) Heatmap showing the L2 distance metric across different configurations.



(b) Heatmap illustrating the Jaccard Distance (One minus Intersection over Union) values for various configurations.

Figure 3.2: Comparison of metrics for unconditional models: (a) L2 distance heatmap and (b) Jaccard heatmap.

does not exhibit the distinct color patterns observed in the paper (Figure 6.3). This led us to explore alternative distance metrics for the kernel.

The derivative of the RBF kernel scales with the pixel-wise differences, $\sim (x_i - x_j)$. Other distance metrics with similar structures, such as Squared Euclidean Distance, Manhattan Distance, and Mahalanobis Distance, would lead to similar outcomes. Additionally, we experimented with the Intersection over Union (IoU) metric. The Jaccard Distance, which intuitively measures the dissimilarity between two sets, was of particular interest. It is based on the ratio of their intersection (shared elements) to their union (combined elements), providing a normalized measure of disjointness. This measure could be more effective in capturing differences between sets.

Using the Jaccard Distance, we propose a new definition for the kernel in Equation 2.11, aiming to improve the performance of particle guidance.

$$d_{IoU}(\mathbf{x}_i, \mathbf{x}_j) = 1 - \text{IoU}(\mathbf{x}_i, \mathbf{x}_j) \tag{3.3}$$

$$\text{IoU}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sum_{k=1}^{d} \min(x_{i,k}, x_{j,k})}{\sum_{k=1}^{d} \max(x_{i,k}, x_{j,k})} = \frac{\text{Intersection}(\mathbf{x}_i, \mathbf{x}_j)}{\text{Union}(\mathbf{x}_i, \mathbf{x}_j)} \tag{3.4}$$

$$\frac{\partial \text{IoU}(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{x}_i} = \frac{\frac{\partial \text{Intersection}(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{x}_i} \cdot \text{Union}(\mathbf{x}_i, \mathbf{x}_j) - \text{Intersection}(\mathbf{x}_i, \mathbf{x}_j) \cdot \frac{\partial \text{Union}(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{x}_i}}{(\text{Union}(\mathbf{x}_i, \mathbf{x}_j))^2} \tag{3.5}$$

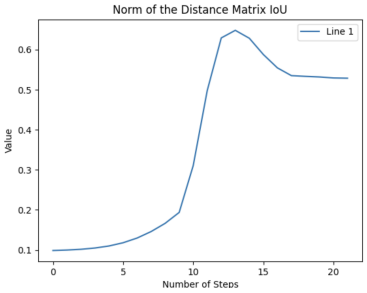
$$\frac{\partial \min(x_{i,k}, x_{j,k})}{\partial x_{i,k}} = \begin{cases} 1 & \text{if } x_{i,k} < x_{j,k}, \\ 0 & \text{if } x_{i,k} > x_{j,k}, \\ \text{undefined (subdifferential)} & \text{if } x_{i,k} = x_{j,k}. \end{cases} \tag{3.6}$$

$$\frac{\partial \max(x_{i,k}, x_{j,k})}{\partial x_{i,k}} = \begin{cases} 1 & \text{if } x_{i,k} > x_{j,k}, \\ 0 & \text{if } x_{i,k} < x_{j,k}, \\ \text{undefined (subdifferential)} & \text{if } x_{i,k} = x_{j,k}. \end{cases} \tag{3.7}$$

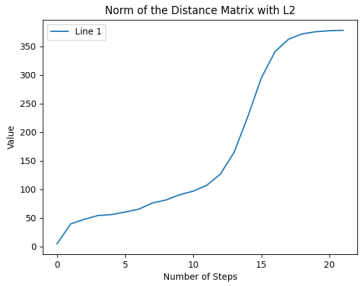
Note that the derivative is not defined for identical values. In such cases, the value is set to zero. The results of using this kernel are shown in Figure 3.2b. These results are an improvement compared to the previous ones, as evident from the presence of bright squares in the heatmap. This indicates that more diverse digits are generated for different configurations. Furthermore, the pattern demonstrates that particle guidance positively impacts diversity, although its influence is not as significant as that of stochastic sampling.

An important factor in the generation process is the quality of the images. The Intersection over Union (IoU) and L2 distance metrics leads to differing image quality. In the appendix, Figure 6.5 shows the generated images for these cases. Using the IoU kernel, the image quality remains stable and acceptable, even in the most challenging case with $S_{\text{churn}} = 1$ and $k_{\text{PG}} = 10$. In contrast, with the L2 kernel, the quality significantly worsens as early as $k_{\text{PG}} = 1$, with artifacts from other samples becoming visible in the generated images.

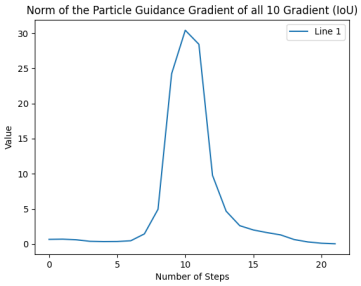
The likely reason for this behavior is the difference in distance values between the two metrics. The distances computed with the L2 norm are much larger than those with IoU since $0 < \text{IoU} < 1$. This is illustrated in Figure 3.3a, where the norm of the distance matrix is computed. The distance metric simply measures the pairwise distances between all elements. Predictably, the distances increase as the denoising steps progress. Initially, the denoiser performs on a noisy image, resulting in a central white mesh, and the diffusion model has not yet determined what to generate (This is visible in the first few steps of the denoising in e.g. Figure 2.5a. At this stage, all 10 images in the batch appear similar, and the distances remain small. As the denoising steps progress and the digits become clearer, the distances grow.



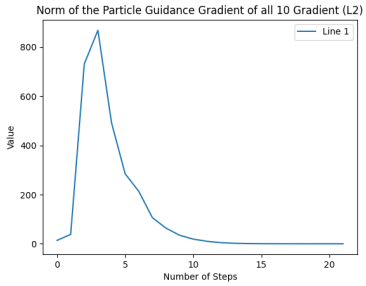
(a) Norm of the distance matrix based on IoU (Intersection over Union) metric of 22 denoising steps.



(b) Norm of the distance matrix based on L2 metric.



(c) Norm of the 10 gradients over time using the IoU norm distance in the kernel.



(d) Norm of the 10 gradients over time using the L2 norm distance in the kernel.

Figure 3.3: Comparison of metrics and gradients: (a) and (b) show the norm of the distance matrix based on IoU and L2 metrics, respectively. (c) and (d) show the norm of 10 gradients over time using L2 and IoU norm distances in the kernel. These plots are gathered with $k_{\text{PG}} = 1$ and $S_{\text{churn}} = 0$.

The issue with the Euclidean norm appears to be that the particle guidance gradient is scaled too strongly, preventing the diffusion model from generating clean images. This is evident when analyzing the particle guidance gradient. Figure 3.3d and Figure 3.3c depict the norm of the gradients, computed over 10 samples. The results show that the L2 gradient norm is significantly higher than that of the IoU. Consequently, the higher gradient magnitudes in the L2 case lead to image corruption, ultimately reducing overall performance. Interestingly, the IoU gradients exhibit a peak in the middle of the process, indicating that they primarily push the particles apart during this stage of the denoising process.

Chapter 4

Conditional Generation

In the previous chapter, we focused on generating different types of digits, where the modes corresponded to the digits themselves. However, our primary interest lies in sequences and how they evolve over time. As described in the introduction, the objective is to predict future frames in a sequence. In this chapter, we detail the methods for conditioning and the training process on the various datasets introduced in section 2.1, followed by the conducted experiments.

We perform two distinct experiments. In the first, we aim to generate different modes, where the modes now represent the next-frame predictions of pixel movements in a stochastic manner. In the second experiment, we evaluate whether the diffusion model can reproduce a specific prior probability in the stochastic movements. For example, if the sequence of images predominantly moves either to the left or the right, we impose a bias by setting the probability of moving to the right to 70% instead of the default 50%.

4.1 Training and Methods

The goal is to condition on previous frames to generate the $t + 1$ frame in the sequence. To achieve this, modifications to the model from the EDM paper are required. While the EDM diffusion model already provides a conditional framework, it is restricted to class embeddings. Many recent diffusion models incorporate text embeddings to generate images described by the user. However, in our case, we aim to use frames as conditions, which eliminates the need for direct embeddings for text or labels.

A straightforward solution is to concatenate the conditional images with the noisy image being generated. This adjustment modifies the network's input channel size to the number of conditional frames plus one, while the output channel size remains one. This approach can be extended to predict multiple future frames by increasing the output channel size accordingly.

The updated denoising equation is now:

$$D_\theta(x, x_{\text{cond}}; \sigma) = c_{\text{skip}}(\sigma)x + c_{\text{out}}(\sigma)F_\theta\left([c_{\text{in}}(\sigma)x, x_{\text{cond}}]; c_{\text{noise}}(\sigma)\right) \tag{4.1}$$

For the first training, we used the Free Moving MNIST dataset, iterating over sequences of length 16, with three conditional frames. In each sequence, a different digit from the 60k dataset was used. This setup allowed the model to encounter two types of scenarios:

1. **Unimode Case**: The digit flows deterministically through the image, representing a single unified mode where the next frame has a unique outcome.
2. **Multimode Case**: The model trains on sequences with multiple possible outcomes for the next frame, requiring it to capture diverse possibilities for the digit's location.

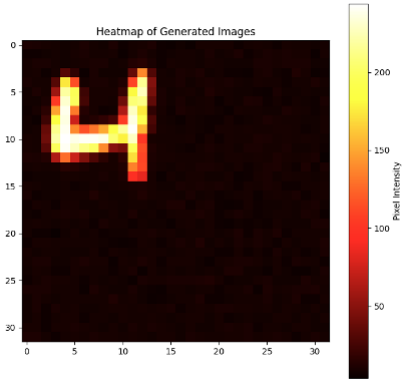
It is crucial that the model generates only a single possible image in the unimode case, even with the stochastic sampler, while in the multimode case, it should produce diverse digit locations.

In Figure 4.1, the inference process using the stochastic sampler is illustrated. For each case, 100 images were generated, and their mean was computed. The results demonstrate that the model successfully distinguishes between the unimode and multimode cases. In the unimode case, the model consistently produces a single outcome, while in the multimode case, it captures a broader variety of directions, as expected.

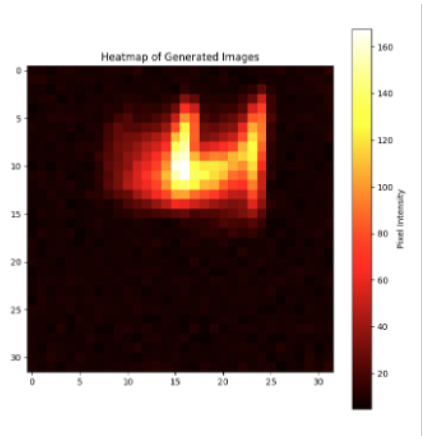
Next, we aim to leverage particle guidance to further explore these scenarios. However, this setup is not suitable for evaluating particle guidance and the stochastic sampler, as it lacks discrete and well-defined modes. Clear modes, such as those illustrated by Gaussian mixtures in Figure 6.3, are necessary for such an evaluation. Therefore, in subsequent experiments, we focus on a simpler multimode scenario to better assess the model's ability to capture diverse outcomes.

For this purpose, we train models on the circle jumping dataset demonstrated in Figure 2.1a. In this dataset, a digit jumps from the center to one of eight equally spaced directions on a circle, each separated by 45°, and all at the same radius. Here, the data loader generates sequences of only two frames: in the first frame, the digit is in the center, and in the second frame, it moves to one of the eight directions. This setup is not quite equivalent to predicting the next frame in a temporal sequence but can instead be interpreted as placing a given digit into one of eight predefined directions.

This results in an eight-mode case, where the goal is to correctly generate all eight directions. This contrasts with the previous experiments, where the



(a) Unimode: The digit flows deterministically through the image, demonstrating single deterministic outcomes. The first three images are the conditional frames. The fourth one is to predict.



(b) Multimode: The model is trained on sequences where multiple outcomes are possible, leading to diverse digit locations. In the sequence, the last conditional frame (of three total) is the second frame with the red marker, encouraging stochastic movement in the next frame prediction.

(c) The sequence used for inference at different time steps.

Figure 4.1: Illustration of inference with the stochastic sampler. (a) Unimode: The mean of 100 generated images, showing deterministic flow. (b) Multimode: The mean of 100 generated images, showing diverse digit locations. (c) The sequence is used for the inference and the conditional frames. Images are grayscale, with pixel values in $x \in [0, 255]$.

model needed to discover all digits in different modes. Using this setup, we aim to evaluate the effectiveness of particle guidance and the stochastic sampler in capturing diverse and well-defined outcomes.

4.2 Exploring stochastic movement

The model is trained to generate the same digit as in the conditional frame and place it randomly in one of the eight possible directions. The objective is to explore whether the model can capture all eight directions within a single batch using the dependent sampler. Training was conducted for 22k update steps with

a batch size of 512.

At inference, the first image from the sequence (as shown in Figure 2.1a) is used as the conditional frame. A batch of eight noisy images is then sampled, and the denoising process, as described in Algorithm 2, is applied to generate the final images. Since the denoiser $D_\theta(\hat{\mathbf{X}}_i; \hat{t}_i)$ was trained on the linear scheduler $\sigma(t) = t$, it is designed to guide the noisy image toward the clean image. Consequently, in high-noise regions, the potential locations for the digit should already be distinguishable, while in low-noise regions, the clear position becomes evident during the denoising process.

This behavior is successfully achieved by the model, as illustrated in the denoising process shown in Figure 6.6.

In the previous chapter, a ResNet was trained to detect the digit for evaluating the outcomes. However, for the purpose of evaluating the direction, a more traditional method can be employed. Specifically, the centroids of the image at the conditional frame t and the generated frame $t+1$ are calculated. The resulting vector is then compared to the eight predefined direction vectors using cosine similarity:

$$\text{Centroid at frame t: } \mathbf{c}_t = \begin{bmatrix} c_1^t \\ c_2^t \end{bmatrix} = \frac{\sum_{i=1}^{H} \sum_{j=1}^{W} \begin{bmatrix} i \\ j \end{bmatrix} \cdot I_t(i,j)}{\sum_{i=1}^{H} \sum_{j=1}^{W} I_t(i,j)} \tag{4.2}$$

$$\text{Direction vector: } \mathbf{v} = \mathbf{c}_{t+1} - \mathbf{c}_t \tag{4.3}$$

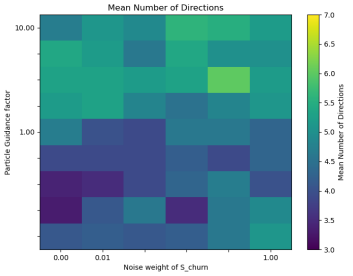
$$\text{Cosine similarity: } y = \arg\max_k S(\mathbf{v}, \mathbf{d}_k) = \arg\max_k \frac{\mathbf{v} \cdot \mathbf{d}_k}{\|\mathbf{v}\| \|\mathbf{d}_k\|}, \quad k = 1, \ldots, 8 \tag{4.4}$$

Previously, ensuring the quality of the generated images was a challenge, and measuring it proved difficult. In this case, however, we have a reference image. By aligning the generated image to its original position, we can calculate the error between them. This allows us to define a quality index for the image:

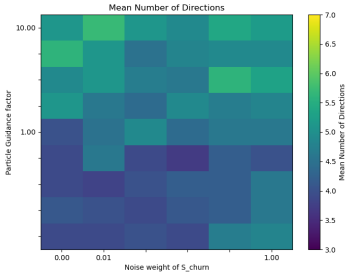
$$\text{Quality score of the image: } \sum_{i=1}^{H} \sum_{j=1}^{W} \|I_t(i,j) - I_{t+1}(i - c_1^{t+1}, j - c_2^{t+1})\|^2 \tag{4.5}$$

In the appendix Figure 6.7 are example images with growing particle guidance factors and the quality score. Looking at those examples. With that one can say the quality score between 7 or 8 the image gets too unreadable to evaluate.

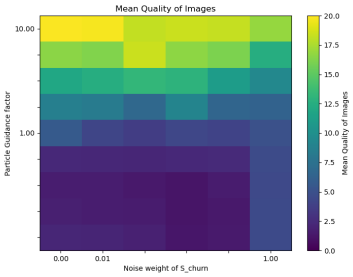
With this in mind, we repeated the experiments using two kernels: the Jaccard Distance and the Euclidean Distance with a negative exponent. In Figure 4.2, when comparing the number of modes discovered, the two kernels do not appear



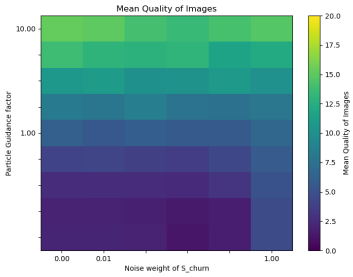
(a) IoU for directions



(b) L2 for directions



(c) IoU for quality



(d) L2 for quality

Figure 4.2: Comparison of metrics for particle guidance. The top row shows the Intersection over Union (IoU) and L2 metrics for directions, while the bottom row presents the same metrics for quality. These plots provide an overview of how particle guidance influences the results across different evaluation dimensions.

to have a significant influence. However, it is noticeable that the patterns become brighter as the values increase.

In terms of image quality, the initial difference seems significant, but if one omits the top three rows of the heatmaps, the results look very similar. The IoU tends to remain dark until the particle guidance factor reaches one. After that, it scales up very quickly. However, it is important to note that once the score exceeds eight, the images are no longer usable. This observation holds true for both kernels. When analyzing the images generated with the L2 kernel at high particle guidance factors, one can observe that it often results in nearly all-black images.

In conclusion, particle guidance appears to slightly improve the discovery of different modes for various positions. Unfortunately, the improvement is minimal. Moreover, the choice of kernel does not seem to play a significant role in achieving diverse positions in the next frame.

In Figure 6.6, it can be seen that the directions for the denoiser are determined at high noise levels. For the next experiments, we decided to explore the scheduler

from the EDM paper. The idea was that it might be beneficial for the denoising process to spend more steps in the high-noise regions. This required training on higher noise levels as well. The scheduler remains linear, but during training, more samples are diffused in the high-noise areas. The noise level σ for each image is sampled from a log-normal distribution, which was changed from the blue distribution to the orange one, as shown in Figure 6.8a.

After training the new model, three schedulers, defined by the parameter $\rho = [7, 4, 3]$, were used to generate results. The behavior of these new schedulers is plotted in Figure 6.8b. Note that $\rho = 7$ is the default setting. In Figure 6.9, the results are compared. It is evident that this modification does not significantly affect the performance of the particle guidance. The reason is that the gradient and its norm from the PG have the greatest effect on the quality and variety of the sampling, the highly noisy images are often suppressed by the PG term.

In conclusion, the effect of particle guidance does not appear to improve significantly with higher scaling factors. However, it still has a noticeable impact on the sampling process, as demonstrated in the next chapter.

4.2.1 Predicting prior probability in movement

For these experiments, the complexity of the movement was reduced. Three different models were trained on the horizontal dataset, where a digit moves horizontally through the image and bounces off the borders. Each time the digit reaches the center, a random direction—either left or right—is sampled. An illustration of this dataset is shown in Figure 2.1c. During dataset creation, we can set the probability for the digit to move in either direction. In the default configuration, this probability is set to 50%. For this experiment, the prior probabilities are set to $p(\text{right}) = [0.5, 0.7, 0.9]$, with $p(\text{left}) = 1 - p(\text{right})$, hence three differently trained models.

This time, we used three conditional frames and trained on the full sequence length of 16. This setup allows the model to learn both unimode and multimode distributions. Other training settings remain unchanged.

At inference time, the goal is to evaluate whether the model can learn the prior probabilities in the multimode case. To do this, we generate a sequence where the last conditional frame t is positioned when the digit is in the center. The two preceding conditional frames, $t - 1$ and $t - 2$, correspond to frames where the digit approaches the center from either the left or the right. This prior movement should not matter, as the direction for the next stochastic frame generation is determined solely by the set probability.

For evaluation, the same approach as before was used. Only this time we use two reference vectors in Equation 4.4. Specifically, 200 sequences, each consisting of a unique digit, were generated. For each sequence, 100 images were created

using different noisy latent spaces. The goal was to assess whether the stochastic and deterministic models could reproduce the prior probabilities. Additionally, we tested if the particle guidance model remains unaffected by the set prior probabilities.

In theory, when the batch size is set to two (matching the number of modes in the multimodal case: left or right), the particles should always push away from each other. This would result in an estimated probability of 0.5 for all trained models.

p(right)	$\hat{p}$(right)		
trained Model	Deterministic	Stochastic	Practical Guidance
0.9	0.9944	0.9598	0.5
0.7	0.7933	0.7294	0.5004
0.5	0.4168	0.5253	0.5006

Table 4.1: Prediction probabilities for different models and types of samplers.

In Table 4.1, the results of the experiments are presented. The rows represent the different trained models, while the columns show the various sampling methods applied to the same trained model. For the stochastic sampler, we set $S_{\text{churn}} = 0.9$, and for particle guidance, we used $S_{\text{churn}} = 0.9$ and $k_{\text{PG}} = 3$, as these settings provided a good trade-off between quality and variety based on the results from the previous section.

First, the deterministic models demonstrate that the original denoising process of the diffusion model struggles to capture the prior probabilities accurately. In the default case with 50%, the predictions underestimate the probability. When there is a shift in the prior, the deterministic model is heavily influenced by this shift in the training data. As noted in the literature on score-based generative modeling, the stochastic sampler introduces a correction term to the denoising process. This behavior is observed here, as the stochastic sampler significantly improves the model's predictions, aligning them more closely with the prior probabilities.

As expected, the sampling method with particle guidance achieved the target 50% for all cases. Notably, the mean quality score of the images was between 3 and 4, indicating that the images were of good quality and not significantly affected by particle guidance. These results suggest that the particles were effectively pushed apart to generate distinct modes in almost every instance.

An impressive observation occurs in the case where the model was trained with a 90% prior probability. The deterministic model generated a direction to the right 99.44% of the time. Despite this, the additional gradient from particle guidance was able to push the samples into the desired modes, even when the gradient of the score function pointed toward another outcome.

Chapter 5

Conclusion

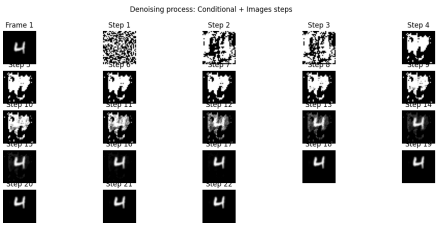
Diffusion models are an excellent method for generating images, but they are heavily influenced by the implementation of the sampler and the training process.

As a conclusion of this project, diffusion models perform exceptionally well in predicting the next frame and likely also subsequent ones. The unimodal case works particularly well since there is only one score guiding the generation process. However, in the multimodal case, diffusion models can be significantly influenced during the generation process. As demonstrated in the last chapter, the model struggled to replicate the exact probabilities for predicting the next frames. This limitation suggests that for more complex applications, such as generating segmentation maps for wildfires or cancer progression rather than a binary left-or-right case, the results should be interpreted with caution, and further research is required.

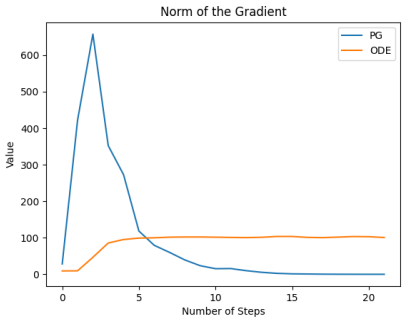
Additionally, it can be summarized that the particle guidance (PG) term, when used with the EDM sampler, works well but perhaps not as effectively as the authors of the original paper might have suggested. The PG term is a valuable adjustment and has shown to perform quite effectively, as supported by the results in Table 4.1. It is particularly useful for discovering all types of modes, presenting a variety of possibilities for scenarios such as predicting where a wildfire or cancer may spread.

On the technical side, the Particle Guidance (PG) term introduces a delicate trade-off between the quality and diversity of the generated images. The root of this issue lies in the gradient magnitudes of the ODE compared to those from PG. For example, in Figure 4.2, which depicts generation from the case with eight possible directions using a single conditional frame, we observe the norm of the particle guidance gradient after scaling, $k_{\text{PG}}(t_i) \cdot \mathbf{d}_{\text{PG}}$, and the vector $\mathbf{d}_i$. These two terms are subtracted in line 10 of the pseudocode 2.

In this specific case, $k_{\text{PG}} = 1.7$, corresponding to the fourth row and left square ($S_{\text{churn}} = 0$) in Figure 4.2 (b) and (d). These configurations lie near the threshold where the diffusion model can still converge to a clean image. The gradient magnitude difference is most evident in the initial denoising step,



(a) Generated images from steps 1 till 22. The upper left images the conditional frame.



(b) Norm of the particle guidance gradient scaled by $k_{\mathrm{PG}}(t_i)$ and compared to the ODE gradient.

Figure 5.1: (a) Displays generated images with eight possible directions using a single conditional frame, while (b) illustrates the gradient norm difference between the particle guidance term, $k_{\mathrm{PG}}(t_i) \cdot \mathbf{d}_{\mathrm{PG}}$, and the vector $\mathbf{d}_i$. These two terms interact in line 10 of the pseudocode (2), and their delicate balance determines whether the model converges to a clean image. It is evident that the gradient norm of the particle guidance term significantly overshoots the norm of the ODE vector at the beginning of the process. This behavior is also visible in the denoising steps of the generated images.

where the Gaussian-distributed noisy image is rapidly transformed into an image influenced by the particle guidance. If the scaling of k_{PG} is too high, the denoiser can no longer converge to a clean image, resulting in poor image quality.

The most significant factors influencing the generation process are the scaling factor k_{PG} and the kernel $k_t(\mathbf{x}_i, \mathbf{x}_j)$. As with many machine learning architectures, it is essential to experiment with hyperparameters and carefully test the functions used to optimize performance.

Chapter 6

Appendix

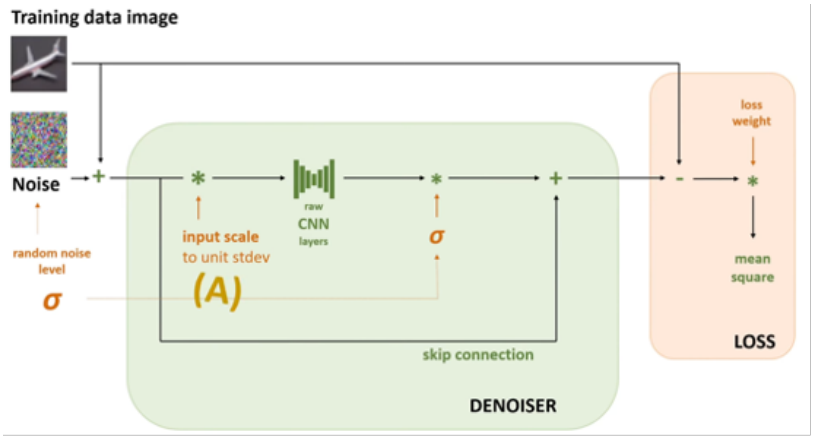
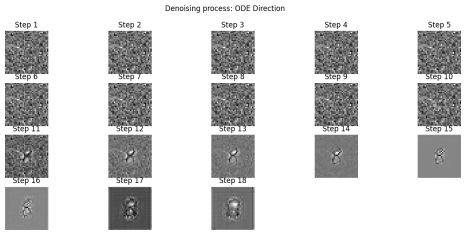
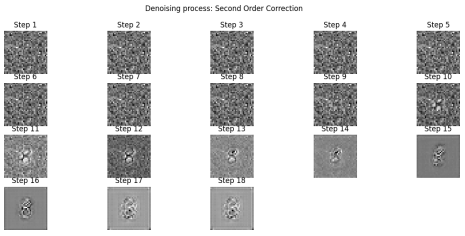


Figure 6.1: This figure provides a clear representation of the model architecture from the EDM paper. Firstly, it is worth noting that the denoising component shown in green corresponds to Equation 4.1, and the loss function is given by Equation 2.5.



(a) Directional visualization of the deterministic ODE, showing the vector field for the denoising process. The d in the Algorithm 1.



(b) Illustration of second-order correction, highlighting the refinement in trajectory alignment. The d'_i in the Algorithm 1.

Figure 6.2: Additional visualizations for the deterministic model: ODE direction (left) and second-order correction (right).

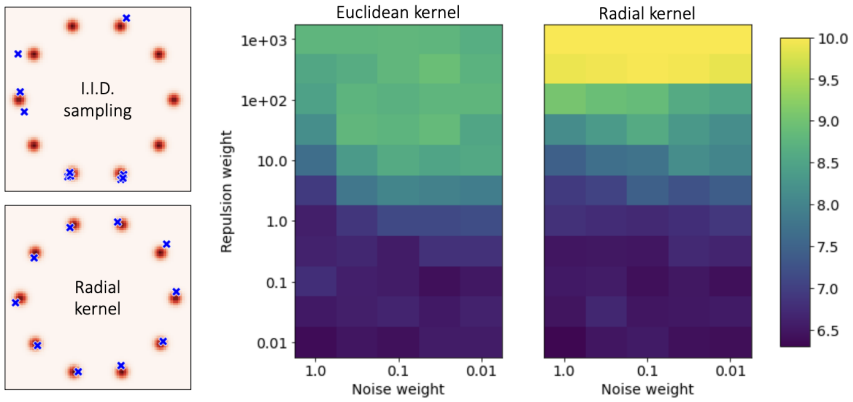
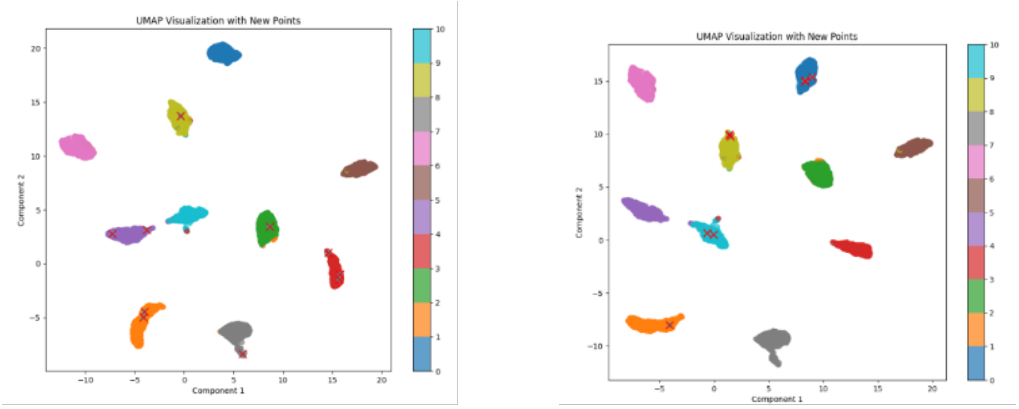


Figure 6.3: Left: plot of random samples (in blue) of the two-dimensional Gaussian mixture distribution (density depicted in red). I.I.D. samples often recover the same modes, while particle guidance with a radial kernel captures all modes. Right: average number of modes recovered with 10 samples as a function of the weight given by the diffusion noising terms and the potential weight when using an RBF kernel with Euclidean and radial distances respectively.



(a) UMAP visualization for particle guidance stage 0, stochastic sampling 1.

(b) UMAP visualization for particle guidance stage 1, stochastic sampling 0.

Figure 6.4: Comparison of UMAP visualizations for different configurations: (a) particle guidance stage 0 with stochastic sampling 1, and (b) particle guidance stage 1 with stochastic sampling 0.

(a) L2 metric visualization for 1 stochastic sampling factor with particle guidance 10.

(b) L2 metric visualization for 0 stochastic sampling factor with particle guidance 1.

(c) IoU metric visualization for 1 stochastic sampling factor with particle guidance 10.

Figure 6.5: Comparison of metrics for unconditional models: (a) L2 metric for stage 1 with particle guidance 10, (b) L2 metric for stage 0 with particle guidance 1, and (c) IoU metric for stage 1 with particle guidance 10.

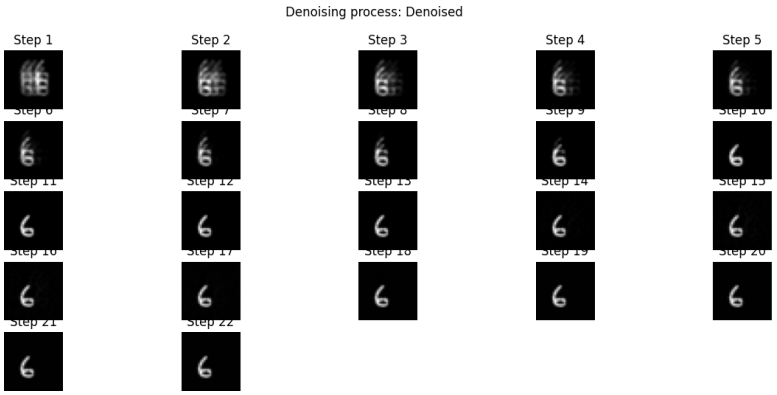
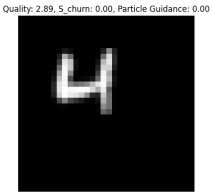
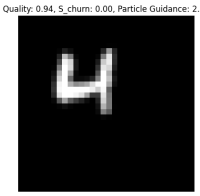


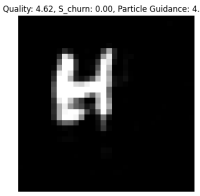
Figure 6.6: Images produced by the denoiser model across the denoising steps. It is evident that the denoiser initially considers eight possible locations for placing the digit '6.' However, it quickly converges to the left side of the image and then to the bottom-left position.



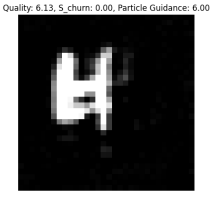
(a) PG 0.0, Quality 2.89



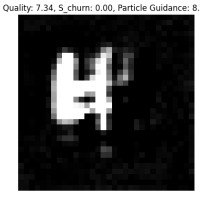
(b) PG 2.0, Quality 0.94



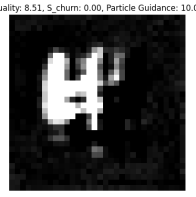
(c) PG 4.0, Quality 4.62



(d) PG 6.0, Quality 6.13

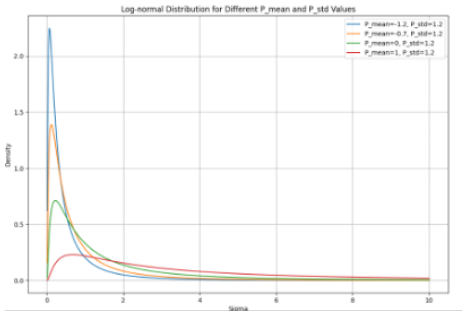


(e) PG 8.0, Quality 7.34

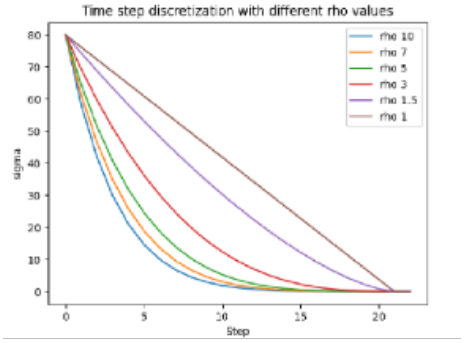


(f) PG 10.0, Quality 8.51

Figure 6.7: Example images with increasing particle guidance (PG) factors and their corresponding quality scores. As the PG factor increases, the quality score also rises, but at scores between 7 and 8, the images become too unreadable to evaluate effectively.

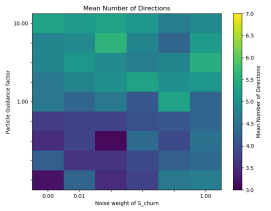


(a) Log-normal distribution used for sampling the noise level σ. The distribution changes from the blue curve to the orange curve to increase **training** focus on high-noise regions.

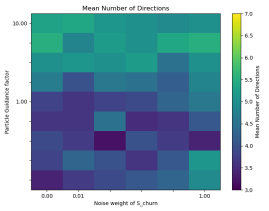


(b) Scheduler behavior defined by $\rho = [1, 1.5, 3, 5, 7, 10]$, showing the influence of different parameter settings during **inference**.

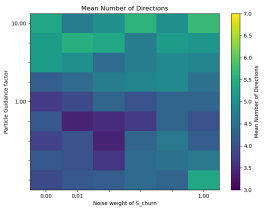
Figure 6.8: Exploration of higher noise training and scheduler behaviors. (a) Log-normal noise distribution adjustment for training, focusing on high-noise regions. (b) Behaviors of schedulers with varying ρ parameters after training on the modified distribution.



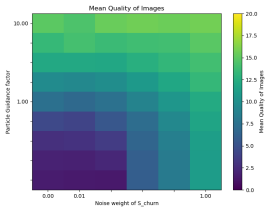
(a) $\rho = 3$: Directional visualization.



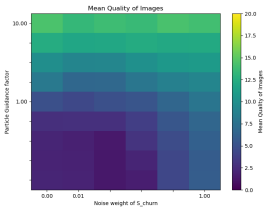
(b) $\rho = 4$: Directional visualization.



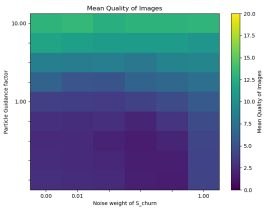
(c) $\rho = 7$: Directional visualization.



(d) $\rho = 3$: Quality scores.



(e) $\rho = 4$: Quality scores.



(f) $\rho = 7$: Quality scores.

Figure 6.9: Results for higher noise training: (top row) directional visualizations for $\rho = 3, 4, 7$ and (bottom row) corresponding quality scores for the same ρ values. The plots demonstrate how the directional behavior and quality scores are influenced by different ρ settings, highlighting the relationship between scheduler parameters and model performance.

Bibliography

[1] G. Corso, Y. Xu, V. de Bortoli, R. Barzilay, and T. Jaakkola, "Particle guidance: non-i.i.d. diverse sampling with diffusion models," 2023. [Online]. Available: https://arxiv.org/abs/2310.13102

[2] T. Karras, M. Aittala, T. Aila, and S. Laine, "Elucidating the design space of diffusion-based generative models," 2022. [Online]. Available: https://arxiv.org/abs/2206.00364

[3] S. Gerard, Y. Zhao, and J. Sullivan, "Wildfirespreadts: A dataset of multi-modal time series for wildfire spread prediction," in *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36. Curran Associates, Inc., 2023, pp. 74 515–74 529. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/file/ebd545176bdaa9cd5d45954947bd74b7-Paper-Datasets_and_Benchmarks.pdf

[4] L. Zbinden, L. Doorenbos, T. Pissas, A. T. Huber, R. Sznitman, and P. Márquez-Neila, "Stochastic segmentation with conditional categorical diffusion models," 2023. [Online]. Available: https://arxiv.org/abs/2303.08888

[5] T. Chen, C. Wang, and H. Shan, *BerDiff: Conditional Bernoulli Diffusion Model for Medical Image Segmentation*. Springer Nature Switzerland, 2023, p. 491–501. [Online]. Available: http://dx.doi.org/10.1007/978-3-031-43901-8_47

[6] A. Rahman, J. M. J. Valanarasu, I. Hacihaliloglu, and V. M. Patel, "Ambiguous medical image segmentation using diffusion models," 2023. [Online]. Available: https://arxiv.org/abs/2304.04745

[7] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," 2021. [Online]. Available: https://arxiv.org/abs/2011.13456

[8] R. Denton and R. Fergus, "Stochastic video generation with a learned prior," 2018. [Online]. Available: https://arxiv.org/abs/1802.07687

[9] N. Srivastava, E. Mansimov, and R. Salakhutdinov, "Unsupervised learning of video representations using lstms," 2016. [Online]. Available: https://arxiv.org/abs/1502.04681

[10] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014. [Online]. Available: https://arxiv.org/abs/1406.2661

[11] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," *Foundations and Trends® in Machine Learning*, vol. 12, no. 4, p. 307–392, 2019. [Online]. Available: http://dx.doi.org/10.1561/2200000056

[12] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," 2015. [Online]. Available: https://arxiv.org/abs/1503.03585

[13] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015. [Online]. Available: https://arxiv.org/abs/1505.04597

[14] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," 2022. [Online]. Available: https://arxiv.org/abs/2010.02502

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. [Online]. Available: https://arxiv.org/abs/1512.03385

[16] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[17] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," 2020. [Online]. Available: https://arxiv.org/abs/1802.03426