ASSIGNMENT III
Generative modeling

SH2150 Machine Learning in Physics

November 2025

Diffusion models have become one of the most exciting ideas in modern generative AI, and at their core is a beautiful interplay between randomness and structure described by stochastic differential equations (SDEs). In this assignment, we dive into that world: from exploring classical processes like Brownian motion and the Ornstein–Uhlenbeck dynamics, to simulating trajectories with Euler–Maruyama, to building neural networks that learn how data is gradually corrupted and then reconstructed. By moving from simple 1D SDEs all the way to UNet-based image models, we get to experience how mathematical theory, numerical simulation, and deep learning come together to form the foundation of today's powerful diffusion and flow-based generative models.

Your tasks are to

- Get familiar with SDEs
- Set up a training/inference pipeline for generative models
- Sample from a 2D distribution using a flow model
- Define a UNet architecture for image generation
- Generate images from a distribution of your choice

1. The theoretical basis for diffusion models is stochastic differential equations (SDE). An SDE is a differential equation symbolically of the form

$$\mathrm{d}X_t = u_t(X_t)\mathrm{d}t + \sigma_t \mathrm{d}W_t, \quad X_0 = x_0, \tag{1}$$

where X_t is the *trajectory*, u_t is the *vector field* or *drift coefficient*, σ_t is the *diffusion coefficient* determining the amount of noise, and W_t is a Brownian motion. The term $\sigma_t \mathrm{d}W_t$ is what gives rise to the stochasticity (randomness) in the system – without it the system reduces to an ordinary differential equation (ODE) $\frac{dX_t}{dt} = u_t(X_t)$.

   (a) Different choices of u_t yield different stochastic processes that can be used to model many different physical phenomena. The Ornstein-Uhlenbeck (OU) process is given by $u_t(X_t) = -\theta X_t,\ \sigma_t = \sigma$, for some constants θ, σ. It can be used to model massive particles under the influence of friction. Show that the OU process is equivalent to Langevin dynamics defined by the drift coefficient (1p)

$$u_t(X_t) = \frac{1}{2}\sigma_t^2 \nabla \log p(X_t),\ \sigma_t = \sigma,$$

when $p(X_t) = \mathcal{N}(0, \frac{\sigma^2}{2\theta})$.

(b) Being able to simulate SDEs will be needed to sample new data points from generative models. The simplest solver for SDEs is the Euler-Maruyama method defined as (1p)

$$X_{t+h} = X_t + hu_t(X_t) + \sqrt{h}\sigma_t\epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, I), \tag{2}$$

where h is the step size. This reduces to the Euler method for ODEs when $\sigma = 0$. Implement the Euler-Maruyama method.

(c) Simulate 10 Brownian motions defined by $\sigma_t = \sigma$, $u_t = 0$, $X_0 = 0$. What happens for different values of σ? (1p)

(d) Simulate the OU process defined above. Run it with a variety of different initial points X_0. Pay attention to the ratio $\frac{\sigma^2}{2\theta}$, and comment on the convergence behavior of the solutions. Are they approaching a particular point or a distribution? (2p)

2. A *diffusion model* is an SDE where the vector field u_t is parametrized by a learnable neural network u_t^θ. If there is no diffusion term, and the SDE is reduced to an ODE, we get a so-called *flow model*.

The network u_t^θ is trained on data points $X_1 \sim p_{\text{data}}$ that have been corrupted with varying amounts of noise corresponding to different time steps X_t, $t \in [0, 1]$, of the SDE/ODE. By simulating the SDE forwards from $t = 0$ to $t = 1$ with our trained vector field, starting with pure noise $X_0 \sim p_{\text{noise}}$, we can generate new data samples from p_{data}.

In a common type of generative model, called denoising diffusion model, the conditional path (which describes how a data point z is corrupted) is given by

$$x_t = \alpha_t z + \beta_t \epsilon, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

where α_t, β_t are continuously differentiable and monotonic noise schedulers with $\alpha_0 = \beta_1 = 0$ and $\alpha_1 = \beta_0 = 1$. This implies $x_t \sim p_t(\cdot \mid z) = \mathcal{N}(\alpha_t z, \beta_t^2 I)$.

(a) Implement linear noise schedulers $\alpha_t = t$, $\beta_t = 1 - t$. (1p)

(b) Implement the conditional path that given a data point z and a time t returns the corrupted data point $x_t = \alpha_t z + \beta_t \epsilon$. (1p)

(c) Define a 2D toy distribution (e.g. a Gaussian mixture) p_{data}. Simulate the corruption process on 1000 samples from p_{data} for $t = 0, 0.25, 0.50, 0.75, 1$, and plot one 2D histogram per time point. (1p)

(d) Modify the noise schedulers so that they are non-linear (while still satisfying the requirements stated above). Then, generate a plot analogous to panel (c) using the same data points z corrupted under this new schedule, and discuss how the change in scheduling influences the corruption process. (1p)

3. There are many ways to train generative models. In this case, we will want to minimize the *conditional flow matching* loss given by

$$L(\theta) = \mathbb{E}_{t\sim U[0,1], z\sim p_{\text{data}}, x\sim p_t(\cdot|z)}[\left\|u_t^\theta(x) - u_t(x \mid z)\right\|^2].$$

(a) The conditional vector field is given by (1p)

$$u_t(x \mid z) = \left(\dot{\alpha}_t - \frac{\dot{\beta}_t}{\beta_t}\alpha_t\right) z + \frac{\dot{\beta}_t}{\beta_t} x.$$

Find a simplified expression for $u_t(x_t \mid z)$ when x_t is drawn from the conditional path $p_t(\cdot \mid z)$, and implement it.

(b) Implement a MLP architecture that takes $(x, t) \in \mathbb{R}^3$ as inputs and outputs the estimated vector field $u_t^\theta(x_t) \in \mathbb{R}^2$. (1p)

(c) Implement a training loop according to Algorithm 1, and train an MLP with 4 hidden layers of dimension 64 on the 2D toy dataset. Choose noise schedulers α_t, β_t according to your liking (that fulfills the needed criteria), and state what you used. Does the loss converge? (1p)

(d) Let the number of time steps $n_t = 1000$, and sample 300 realizations from p_{data} using the Euler method ($\sigma_t = 0$) applied to the ODE parametrized by your trained vector field. Plot a 2D scatter plot of 300 corrupted data points according to the true conditional path for $t = 0, 0.25, 0.50, 0.75, 1$. Provide a similar plot for the 300 generated data points together with a plot of the simulated trajectories X_t. How does the choice of n_t affect the sampling? (1p)

Algorithm 1: Conditional flow matching

Require: data set with samples p_{data}, vector field u_t^θ
for *each batch* **do**
  Sample data point $z \sim p_{\text{data}}$.
  Sample random time $t \sim U[0, 1]$.
  Sample noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
  Set $x_t = \alpha_t z + \beta_t \epsilon$.
  Compute loss $L(\theta) = \left\| u_t^\theta(x_t) - u_t(x_t \mid z) \right\|^2$.
  Update model parameters θ using gradient step on $L(\theta)$.
end

4. Let us now turn to image generation. To handle high-dimensional image data, we need another architecture than an MLP to parameterize our vector field. We will use the famous UNet architecture with some modification to allow for the time embedding (the network must be informed about the current time t and for CNNs it is not as straightforward to do this as for MLPs where we just fed it as an additional input).

(a) Select a few data points z from your choice of image data set, and provide a plot of the corresponding corrupted data points x_t for time points $t = 0, 0.25, 0.50, 0.75, 1$. (1p)

(b) Implement the residual layer of the UNet architecture and add the pre-defined time embedding according to figure 1. Train a vector field over 5000 epochs on the image data with batch size of 250 using your UNet. (3p)

(c) Sample a couple of images from the image distribution by solving the flow ODE $dX_t = u_t^\theta(X_t)dt$ while using your trained UNet as the drift term. (1p)

(d) Play around with different noise schedulers and comment on how they affect the sample quality. (1p)

5. So far we have only trained flow models (no diffusion term). Via the Fokker-Planck equation, it can be shown that the following SDE

$$dX_t = \left(u_t^\theta(X_t) + \frac{\sigma_t^2}{2} \nabla \log p_t(X_t) \right) dt + \sigma_t dW_t \tag{3}$$

have the same probability paths p_t as the flow ODE $\frac{dX_t}{dt} = u_t^\theta(X_t)$. Hence, to convert the flow model into a diffusion model, we have to learn the second drift term $\nabla \log p_t(X_t)$ called the *score function*. This can be done by training a network s_t^θ using conditional score matching

$$L(\theta) = \mathbb{E}_{t \sim U[0,1], z \sim p_{\text{data}}, x \sim p_t(\cdot|z)} [\left\| s_t^\theta(x) - \nabla \log p_t(x \mid z) \right\|^2].$$

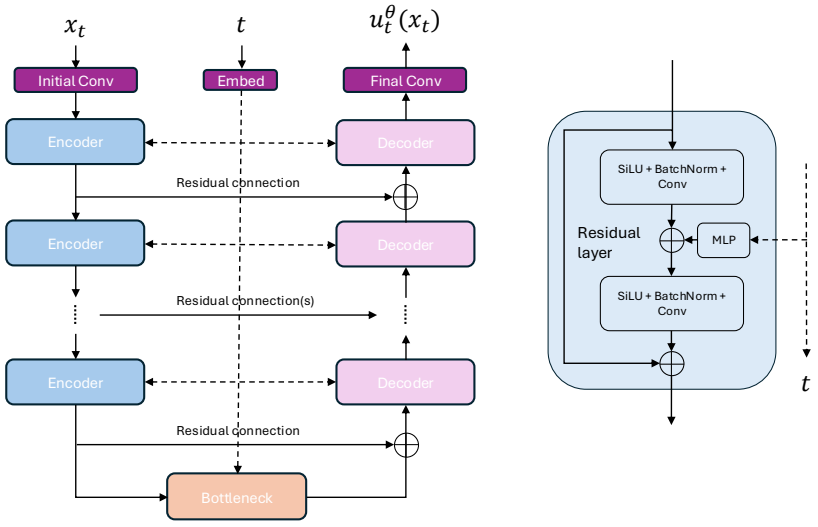

Figure 1: UNet architecture together with a detailed overview of the residual layer. Each encoder consists of two residual layers followed by downsampling. Each decoder consists of upsampling followed by two residual layers. The bottleneck module consist of three residual layers.

(a) Derive an explicit formula for the conditional score function $\nabla \log p_t(x \mid z)$ from the conditional path and implement it. Then train a score network s_t^θ on the image dataset using the conditional score matching loss. (3p)

(b) Sample new data using the diffusion model by combining the trained vector field u_t^θ and the score network s_θ^t and simulate the SDE in (3). How does the value of σ_t affect the samples? (3p)

Final question: Did you use an AI tool (other than the machine learning models you trained in this exercise) for anything else than information searching, when solving this problem set? If so, please write a **brief statement of how you used AI.**

Total number of points: 25

Motivate your answers wherever applicable.

Good luck!