ETH zürich

# SEMESTER PAPER

## Diffusion Models: theoretical background and a case study with stochastic localisation

**Edoardo Tarcisio Visconti**

Supervisor: Yuansi Chen

Spring 2025

# Contents

# 1 Introduction

Diffusion models have emerged as a powerful framework for generative modeling, achieving state-of-the-art results in tasks such as image synthesis and density estimation. These models are typically formulated in a probabilistic setting, where the goal is to learn a generative process that, given i.i.d. samples from an unknown target distribution $\mu$, can generate new samples from the same distribution. Diffusion models achieve this by learning the score function, i.e., the gradient of the log-likelihood.

After briefly comparing diffusion models with other approaches used in generative modeling, we will introduce methods for learning the score function, such as score matching and denoising score matching (along with references to other techniques). Once the score function is learned, we will explore various sampling algorithms, including *annealed Langevin dynamics* [6] (by Song and Ermon), *denoising diffusion probabilistic models (DDPM)* [12] (by Ho et al.), and *stochastic localization* [5] (by Montanari).

Then we will analyze key results from the paper by Montanari [5], which provides further insights into the empirical properties of these methods. In the end, following the approach of Shah et al. [13], we study the convergence properties of stochastic localization when applied to a two-component Gaussian mixture model.

# 2 Different methods for generative modeling

Generative modeling aims to learn a data distribution $\mu$ from i.i.d. samples and generate new samples from the same distribution. Three major approaches can be distinguished: likelihood-based models, implicit generative models, and score-based models.

- **Likelihood-based models** explicitly define a parametric probability density function $p_\theta(x)$ and learn the parameters $\theta$ via (approximate) maximum likelihood estimation (MLE). That is, they assume

$$\mu(x) \propto p_\theta(x),$$

  where $p_\theta(x)$ is a probability density function parameterized by $\theta$. The parameters are optimized by maximizing the log-likelihood of the observed data,

$$\max_\theta \sum_{i=1}^{N} \log p_\theta(x_i).$$

  This category includes autoregressive models, variational autoencoders (VAEs) and others. While likelihood-based models offer a well-defined probabilistic framework and direct likelihood evaluation, they often require computationally expensive normalizing constraints or variational approximations.

- **Implicit generative models** do not explicitly define a probability density function. Instead, they represent the data distribution as a transformation of a simple prior distribution, typically a Gaussian:

$$g(Z), \quad Z \sim \mathcal{N}(0, I_m),$$

where $g : \mathbb{R}^m \to \mathbb{R}^n$ is a learnable mapping. Instead of modeling $p_\theta(x)$ explicitly, these methods define a generative process and learn $g$ by minimizing a divergence between two probability measures $\mathcal{D}_{\text{distributional}}(\cdot, \cdot)$:

$$\mathcal{D}_{\text{distributional}}(g(Z), \frac{1}{N}\sum_{i=1}^{N} \delta_{x_i}(\cdot))$$

where $\frac{1}{N}\sum_{i=1}^{N} \delta_{x_i}(\cdot)$ represents the empirical data distribution. The most well-known example of implicit generative models is generative adversarial networks (GANs) [2], where new samples are synthesized by transforming a random Gaussian vector using a neural network $g$. The parameters of $g$ are learned by minimizing an adversarial loss between generated and real samples. Implicit models can capture highly complex distributions without requiring explicit likelihood estimation but often suffer from challenges such as training instability and mode collapse.

- **Score-based models** take a different approach by estimating the *score function*, i.e., the gradient of the log-density,

$$s_\theta(x) = \nabla_x \log p_\theta(x).$$

Instead of directly modeling $p(x)$, score-based methods learn $s_\theta(x)$ from data and use it for generative sampling. This is typically done via stochastic differential equations (SDEs) or iterative refinement techniques such as Langevin dynamics. Since these methods do not require explicit likelihood estimation, they avoid the computational burden of normalizing constants and are particularly effective in high-dimensional settings. Score-based models form the foundation of modern diffusion models, where the score function is learned across multiple noise scales, defining a structured generative process, like in [6] and [12].

Each of these approaches comes with its own strengths and limitations. Likelihood-based models offer direct density estimation but may struggle with normalization constraints. Implicit generative models provide flexibility but require adversarial training, which can be unstable. Score-based models bypass explicit likelihood estimation but depend on accurate score estimation and numerical solvers for sampling.

As previously mentioned, this project will focus on the third approach and will explore methods for learning the score function and using it in generative processes.

# 3 Score function and score matching

## 3.1 Score function and Langevin dynamics

In the machine learning literature, the term *score function* refers to the gradient of a probability density $\mu$, which is often the likelihood of an unknown model. We define it as:

$$s(x) = \nabla_x \log \mu(x).$$

As discussed by Song and Ermon in [6], the score function can be interpreted as a vector field that describes how the density $\mu(x)$ varies across different regions of space, always pointing in the direction of increasing probability. Heuristically, it provides a way to compare the likelihood of a sample $x$ with its neighboring points.

If the score function is known, one can generate samples from the target distribution $\mu$ using the Langevin stochastic differential equation (SDE):

$$dX_t = s(X_t)dt + \sqrt{2}dW_t, \quad X_0 \sim \pi_0,$$

where $W_t$ is a standard Wiener process and $\pi_0$ is an initial prior distribution, typically chosen as a simple Gaussian. Under mild assumptions on $\mu$, this SDE converges to its stationary distribution $\mu$, ensuring that long-run samples follow the target measure.

In practice, this continuous process is approximated via discretization using the Euler-Maruyama method, leading to the unadjusted Langevin algorithm (ULA):

$$X_{k+1} = X_k + \eta_k s(X_k) + \sqrt{2\eta_k} Z_k, \quad Z_k \sim \mathcal{N}(0, I).$$

where $\eta_k$ is the timestep $k^{th}$. For details about ULA, see Chapter 6.3.3 of [11].

The answer we have yet to answer is how to estimate the score function when the target measure $\mu$ is only available through i.i.d. samples (i.e. in the setting of generative modelling).

## 3.2 Score matching

Score matching, first introduced by Hyvarinen in [3] provides a way to estimate the score function without explicitly computing the density $\mu(x)$, making it particularly useful in our setting. In particular minimizing the expected squared error between the true score function $\nabla_x \log \mu(x)$ and an estimated function $s_\theta(x)$ (which in practice will be a neural network) has an equivalent formulation without the need of specifying the actual score function:

$$\arg\min_\theta \mathbb{E}_{X\sim\mu}\left[\|\nabla_x \log \mu(X) - s_\theta(X)\|_2^2\right] = \arg\min_\theta \mathbb{E}_{X\sim\mu}\left[\|s_\theta(X)\|^2 + 2\nabla_x \cdot s_\theta(X)\right]. \tag{1}$$

and for computing purposes we replace the expectation with the empirical average over our i.i.d. samples, yielding the practical score matching objective:

$$\min_\theta \frac{1}{N}\sum_{i=1}^{N}\left[\|s_\theta(x_i)\|^2 + 2\nabla_x \cdot s_\theta(x_i)\right].$$

and finally we use autodifferentiation to compute the divergence.

We follow [3] to prove the equivalence. First, we expand the squared norm:

$$\mathbb{E}\|\nabla_x \log \mu(X)\|_2^2 - 2\mathbb{E}\langle s_\theta(X), \nabla_x \log \mu(X)\rangle + \mathbb{E}\|s_\theta(X)\|^2.$$

Now, since the first term does not depend on $\theta$, minimizing the objective is equivalent to minimizing:

$$-2\mathbb{E}\langle s_\theta(X), \nabla_x \log \mu(X)\rangle + \mathbb{E}\|s_\theta(X)\|^2.$$

Then we rewrite the first expectation in integral form and use the divergence theorem (assuming $\mu(x)$ has sufficient smoothness and noticing it vanishes at infinity):

$$\begin{aligned}
\mathbb{E}_{X\sim\mu}\langle s_\theta(X), \nabla_x \log \mu(X)\rangle &= \int \langle s_\theta(x), \nabla_x \log \mu(x)\rangle \mu(x) dx \\
&= -\int \nabla_x \cdot \big(s_\theta(x)\mu(x)\big) dx \\
&= -\mathbb{E}_{X\sim\mu} \nabla_x \cdot s_\theta(X).
\end{aligned}$$

Thus, concluding the proof.

## 3.3 Limitations of ULA for generative modeling

The sampling algorithm we have discussed so far can be seen as a prototype of a diffusion model. However, it is not practical due to the limitations of both ULA and Score Matching.

In particular, Song and Ermon [6] (Section 3.2) demonstrated that Score Matching struggles in low-density regions, i.e., areas far from the modes of the data distribution where fewer samples are available. Heuristically, ULA can be thought of as a process that gradually *moves samples from an initial distribution toward the target distribution*. If the score function is poorly estimated in low-density regions, where the initial distribution has high probability mass, ULA will fail to guide samples correctly toward the true distribution.

Another drawback of Score Matching is that it requires computing the divergence of the estimated score function, which involves calculating $n$ partial derivatives. This becomes computationally prohibitive for high-dimensional data, such as images, where the dimensionality is in the order of thousands or millions.

Furthermore, Langevin-based algorithms can suffer from slow mixing, particularly in the presence of complex multi-modal distributions. When the data distribution exhibits multiple well-separated modes, the algorithm struggles to transition between them efficiently. If initialized in a single mode, Langevin dynamics may take an exponentially long time to reach another mode. Even if initialized across multiple modes, it may not correctly estimate the relative weights of these modes within a reasonable number of iterations, leading to biased sampling in high-dimensional spaces.

These limitations highlight the need for improved generative modeling approaches that enhance sampling efficiency and score function estimation, motivating the development of Denoising Score Matching and annealed Langevin Dynamics.

# 4 DSM and Annealed Langevin Dynamics

## 4.1 Denoising Score Matching

Denoising Score Matching (DSM) was introduced by Vincent in [8] to improve the estimation of the gradient of the log-density in generative models. As discussed in the previous section, a major limitation of standard score matching is its poor performance in low-density regions, where gradient estimation becomes inaccurate. DSM addresses this issue by adding controlled noise to the data (by convolving $\mu$ with Gaussian noise) and training the model to predict the gradient of the resulting noisy distribution.

Specifically, given a perturbation kernel $q_\sigma(\tilde{x}|x)$, the perturbed data distribution is defined as:

$$\mu_\sigma(\tilde{x}) = \int \mu(x) q_\sigma(\tilde{x}|x) dx.$$

DSM minimizes the following loss:

$$\mathbb{E}_{X\sim\mu}\mathbb{E}_{\tilde{X}\sim q_\sigma(\tilde{x}|x)}\left[\|s_\theta(\tilde{X}) - \nabla_{\tilde{x}} \log q_\sigma(\tilde{X}|X)\|_2^2\right]. \tag{2}$$

As shown in [8], this is equivalent to minimizing the original problem (under some regularity conditions):

$$\mathbb{E}_{X\sim\mu}\mathbb{E}_{\tilde{X}\sim q_\sigma(\tilde{x}|x)}\left[\|s_\theta(\tilde{X}) - \nabla_{\tilde{x}} \log \mu_\sigma(\tilde{X})\|_2^2\right].$$

We will have (mostly) full control over the perturbation kernel, and it is common to choose:

$$q_\sigma(\tilde{x}|x) = \mathcal{N}(\tilde{x}; x, \sigma^2 I).$$

whose score function is

$$\nabla_{\tilde{x}} \log q_\sigma(\tilde{x}|x) = \frac{1}{\sigma^2}(\tilde{x} - x).$$

and the new target density is therefore defined as:

$$\mu_\sigma(\tilde{x}) := \int \mu(x) q_\sigma(\tilde{x}|x)\, dx = \mu * \mathcal{N}(0, \sigma^2 I).$$

Using the reparameterization trick, we obtain:

$$\tilde{X} = X + \sigma Z, \quad \text{where } X \sim \mu, \quad Z \sim \mathcal{N}(0, I).$$

and, for a single noise level $\sigma$, the training objective becomes:

$$\min_\theta \mathbb{E}_{X\sim\mu, \tilde{X}\sim q_\sigma(\tilde{x}|x)}\left[\left\| s_\theta(\tilde{X}) + \frac{1}{\sigma^2}(\tilde{X} - X)\right\|^2\right].$$

which is equivalent to:

$$\min_{\theta} \mathbb{E}_{X\sim\mu, Z\sim\mathcal{N}(0,\sigma^2 I_n)} \left\| s_\theta(X + \sigma Z) + \frac{Z}{\sigma} \right\|_2^2 . \tag{3}$$

This formulation can then be easily computed using its empirical counterpart.

Notably, with DSM, the objective has shifted from directly modeling the data density (via its score) to learning how to denoise the data from Gaussian noise.

Clearly, the equality $s_\theta(x) = \nabla_x \log \mu_\sigma(x) \approx \nabla_x \log \mu(x)$ holds only when the noise is small enough. A natural way to extend this framework for generative modeling is to study the score function across different noise levels $\sigma$, such that the learned scores can be leveraged to define a structured generative process. This is usually achieved by training a shared neural network across different noise levels, commonly referred to as a Noise Conditional Score Network (NCSN), as introduced in [6], by finding:

$$\theta^* = \arg\min_{\theta} \sum_{i=1}^{N} \sigma_i^2 \mathbb{E}_{X\sim\mu} \mathbb{E}_{\tilde{X}\sim q_{\sigma_i}(\tilde{x}|x)} \left[ \| s_\theta(\tilde{X}, \sigma_i) - \nabla_{\tilde{x}} \log q_{\sigma_i}(\tilde{X}|X) \|_2^2 \right].$$

In particular, we start with a large $\sigma$ so that the distribution is dominated by Gaussian noise, and progressively decrease $\sigma$ to guide samples toward the data distribution $\mu$. Specifically, by considering a progressively decreasing noise schedule, we can iteratively refine samples towards the data distribution.

This motivates the discussion of *Annealed Langevin Dynamics*, a method that incorporates a multi-scale approach to sampling by progressively reducing the noise variance and using Langevin updates to reconstruct high-quality samples.

## 4.2 Annealed Langevin Dynamics (ALD)

In this section, we closely follow the work of Song and Ermon, [6], which introduces ALD.

The key prerequisite for ALD is the training of a NCSN, as explained in the previous paragraph, to estimates the score function $s_\theta(x, \sigma) = \nabla_x \log \mu_\sigma(x)$ for different noise levels $\sigma$.

Once the NCSN is trained, ALD can be used to generate samples by initializing from a simple prior distribution (typically a high-variance Gaussian) and applying Langevin updates while gradually decreasing the noise level.

The resulting procedure is the following:

1. **Train an NCSN** to estimate $s_\theta(x, \sigma)$ for multiple noise levels $\{\sigma_t\}_{t=1}^T$ such that $\sigma_1 > \sigma_2 > \cdots > \sigma_T$.
2. **Initialize** samples from a Gaussian prior $x_0 \sim q_T(x) = \mathcal{N}(0, I)$.
3. **For** $t = T$ down to 1:

(a) Define the step size as

$$\alpha_t = c \cdot \frac{\sigma_t^2}{\sigma_T^2}. \tag{4}$$

(b) Perform one iteration of Langevin Dynamics:

$$x_{t+1} = x_t + \alpha_t s_\theta(x_t, \sigma_t) + \sqrt{2\alpha_t} Z_t, \quad Z_t \sim \mathcal{N}(0, I). \tag{5}$$

4. **Return** the final samples $x_T$.

This process ensures that the initial samples, dominated by Gaussian noise, explore the entire space, avoiding the curse of dimensionality. As the noise level $\sigma_t$ decreases, the samples gradually move toward the true data distribution.

The performance of ALD depends critically on the choice of noise schedule $\{\sigma_t\}$ and step size $\alpha_t$. Empirical tuning and adaptive noise scheduling can improve performance, ensuring that ALD produces accurate and diverse samples.

## 4.3 DDPMs

Denoising Diffusion Probabilistic Models (DDPM), described by Ho et al. [12], can be viewed as a discretized and parameterized variant of ALD. In particular, DDPM replaces the iterative Langevin procedure with a fixed-length Markov chain where the forward process progressively corrupts data points $x_0 \sim p_{\text{data}}(x)$ via:

$$q(x_i \mid x_{i-1}) = \mathcal{N}\left(x_i; \sqrt{1-\beta_i} x_{i-1}, \beta_i I\right),$$

with noise schedule $\{\beta_i\}_{i=1}^N$ and corresponding $\alpha_i := \prod_{j=1}^i (1-\beta_j)$. The marginal distribution conditioned on $x_0$ is:

$$q(x_i \mid x_0) = \mathcal{N}\left(x_i; \sqrt{\alpha_i} x_0, (1-\alpha_i) I\right),$$

and the perturbed data distribution is:

$$q_{\alpha_i}(\tilde{x}) := \int p_{\text{data}}(x)\, q(x_i = \tilde{x} \mid x)\, dx.$$

As before, the reverse process is learned using a neural network $s_\theta$ that approximates the score function of the conditional perturbation kernel. The model is trained by minimizing a weighted denoising score-matching objective (a reweighted variant of the ELBO):

$$\theta^* = \arg\min_\theta \sum_{i=1}^N (1-\alpha_i)\, \mathbb{E}_{x_0 \sim p_{\text{data}}} \mathbb{E}_{x_i \sim q(x_i \mid x_0)} \left[ \left\| s_\theta(x_i, i) - \nabla_{x_i} \log q(x_i \mid x_0) \right\|^2 \right].$$

After training, samples can be generated by starting from $x_N \sim \mathcal{N}(0, I)$ and applying the learned reverse process iteratively:

$$x_{i-1} = \frac{1}{\sqrt{1-\beta_i}} \left(x_i + \beta_i s_{\theta^*}(x_i, i)\right) + \sqrt{\beta_i} z_i, \quad z_i \sim \mathcal{N}(0, I).$$

This is known as *ancestral sampling*, since it samples from the reverse Markov chain defined by $\prod_{i=1}^N p_\theta(x_{i-1} \mid x_i)$.

# 5 Score based generative modeling through SDEs

This section and the next one closely follow the works of Song et al. [7] and Montanari [5]. Previous methods have successfully modeled data by perturbing it with multiple noise levels in a discrete-time framework. In particular, Song et al. [7] and Ho et al. [12] describe DDPMs as discrete-time Markov chains. This Markovian structure arises because both the forward diffusion process and the reverse denoising process depend only on the previous time step and are independent of earlier states. A natural extension of this approach is to generalize the discrete-time formulation to a continuous-time setting with an infinite number of noise scales. In this framework, the perturbed data evolves according to a Markovian SDE (such as the OU process) instead of a discrete-time Markov chain. As we will see later, score matching plays a crucial role in this formulation as well.

## 5.1 Perturbing data with SDEs and reversing the process

Given a fixed $T > 0$, we construct a diffusion process $X = \{X_t\}_{t\in[0,T]}$ and denote by $p_t$ the marginal distribution of $X_t$ and $p_{0t}$ the conditional distribution of $X_t$ knowing $X_0$. We aim to define a process such that $X_0 \sim p_0 = \mu$, the target distribution we want to sample from and $X_T \sim p_T$, a distribution that is easy to sample from, e.g., standard Gaussian noise.

We model the diffusion process using an Itô SDE:

$$dX_t = F(X_t, t)dt + g(t)dW_t,$$

where

- $W_t$ is a $d$-dimensional Brownian motion
- $F(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^d$ is a deterministic vector-valued function (the drift coefficient)
- $g(\cdot) : \mathbb{R} \to \mathbb{R}$ is a deterministic scalar function (the diffusion coefficient).

If we can reverse this process by starting with samples $X_T \sim p_T$, we can obtain samples $X_0 \sim p_0$. Following the notation of Montanari [5], we define the time-reversed process as:

$$\bar{X}_t = X_{T-t}.$$

A notable result by Anderson (1982) states that the reverse of a diffusion process is still a diffusion process:

$$d\bar{X}_t = \bar{F}(\bar{X}_t, t)dt + \bar{g}(t)d\bar{W}_t,$$

where:

$$\bar{F}(\bar{X}_t, t) = -F(\bar{X}_t, T-t) + g(T-t)^2\nabla_x \log p_{T-t}(\bar{X}_t),$$
$$\overline{g}(t) = g(T-t).$$

With suitable modifications, this approach works for any appropriate time transformation $s(t)$ (see [5], Section 1.2 for details).

## 5.2 Estimating the score for the SDE with DSM

It is now clear that to obtain samples from $\mu$, one has to simulate the reverse process. However, prior to that, it is crucial to estimate the score function $\nabla_x \log p_t(x)$. Since the true score function is generally unknown in practical applications, we approximate it by minimizing the DSM objective over different values of $t$ (see (2), but the noise is indexed by $t$):

$$\theta^* = \arg\min_\theta \mathbb{E}_t \left\{ \lambda(t) \mathbb{E}_{X_0 \sim \mu(x_0)} \mathbb{E}_{X_t \sim p_{0t}(x_t|x_0)} \left[ \| s_\theta(X_t, t) - \nabla_{x_t} \log p_{0t}(X_t|X_0) \|_2^2 \right] \right\} \quad (6)$$

and $\lambda(t) : [0, T] \to \mathbb{R}_0$: is a positive weighting function, and t is in practise sampled uniformly over $[0, T]$. Eventually, the method is not yet ready to be used: to solve (6) we need to know $p_{0t}(x_t|x_0)$. Fortunately, if we choose $F(\cdot, t)$ affine, the transition kernel is a Gaussian distribution with known mean and variance.

With this, we now have all the necessary ingredients to obtain samples from $p_0$ by simulating the reverse process. This can be done using general-purpose SDE solvers such as Euler-Maruyama or Stochastic Runge-Kutta methods. Alternatively, Song et al. in [7] propose more advanced approaches, such as Predictor-Corrector (PC) samplers, which combine an SDE solver (predictor) with a Langevin MCMC step (corrector) to refine sample quality, and denoising diffusion implicit models (DDIM), which accelerate sampling by solving an ODE instead of the full stochastic process.

If the estimation of $p_{0t}$ is prohibitive, instead of the DSM objective (6), one can use other techniques to approximate the score, such as Sliced score matching (which is not discussed here but can be found in [14]).

## 5.3 An example with OU process

An example of such a process is the standard Ornstein-Uhlenbeck (OU) process, (with mean reversion rate $\theta = 1$ and long-term mean $\mu = 0$):

$$dX_t = -X_t dt + \sigma dW_t,$$

for some fixed $\sigma > 0$. Of such process, we know its transition kernel,

$$p_{0t}(x_t|x_0) = \mathcal{N}\left( e^{-t} x_0, \frac{\sigma^2}{2} \left(1 - e^{-2t}\right) I \right).$$

i.e. $X_t = e^{-t} X_0 + \sqrt{\frac{\sigma^2}{2}(1 - e^{-2t})} Z$ for $Z \sim \mathcal{N}(0, I)$.

Morever, as $T \to \infty$, the process converges to its stationary distribution (whatever the starting point):

$$p_T(x) = \mathcal{N}(0, I).$$

In a nuthsell, the OU process transforms samples from $\mu$ into samples of pure Gaussian noise. In this case, with reparameterization and computing the gradient, (6) becomes:

$$\theta^* = \arg\min_\theta \mathbb{E}_t \left\{ \lambda(t) \mathbb{E}_{X_0 \sim \mu(x_0)} \mathbb{E}_{Z \sim \mathcal{N}(0,I)} \left[ \left\| s_\theta(X_t, t) + \left( \frac{\sigma^2}{2} \left(1 - e^{-2t}\right) \right)^{-\frac{1}{2}} Z \right\|_2^2 \right] \right\} \quad (7)$$

which is easily substituted by its numerical counterpart. To sum up, we sample with the following SDE (with $\sigma^2 = 2$):

$$dX_t^{\leftarrow} = \left[\bar{X}_t + 2\nabla \log q_{T-t}\left(X_t^{\leftarrow}\right)\right] dt + \sqrt{2}\, dB_t$$

which we approximate with

$$dX_t^{\leftarrow} = \left[\bar{X}_t + 2s_\theta\left(X_t^{\leftarrow}, T-t\right)\right] dt + \sqrt{2}\, dB_t$$

and it is discretized as:

$$X_{t+1}^{\leftarrow} - X_t^{\leftarrow} = \left[X_t^{\leftarrow} + 2s_\theta\left(X_t^{\leftarrow}, T-t\right)\right] \Delta t + \sqrt{2\Delta t} Z_i.$$

# 6 Stochastic localization (a special case)

Inspired by [5], in this section, we present the ideas behind stochastic localisation processes. To do so, we will present a special case (using Gaussian noise again) and see that it is connected to subsection **??**. Yet, a general theory of general stochastic localisation sampling exists and, along with various different formulations, is presented in the same work.

A general stochastic localization process is a process $(\mu_t)_{t\geq 0}$ taking values in probability distributions on $\mathbb{R}^n$ (this means that at each time t we are given a random measure $\mu_t$) satisfying:

- As $t \to \infty$, $\mu_t$ "localizes" to a degenerate distribution $\delta_{\mathbf{x}_*}$.
- $\mu_t$ is a martingale.

However, a natural alternative interpretation, instead of working with probability measures, is to consider a noisy observation $Y_t$ of a latent variable $\mathbf{x}_* \sim \mu$ such that it becomes more informative as t increases. Importantly, $Y_t$, the observation process, does not need to be in the same space as $\mathbf{x}_*$. Then we model $u_t(x \in \cdot) = \mathbb{P}(x \in \cdot | Y_t)$.

In our example, $Y_t$ follows a linear noisy model:

$$Y_t = t\mathbf{x}_* + W_t,$$

where $W_t$ is a standard Brownian motion, and it is clear that the signal-to-noise ratio increases as t increases. Using the Bayes rule, we compute the conditional distribution:

$$\mu_t(dx) \propto \mu(dx) \exp\left\{-\frac{1}{2t}\|Y_t - tx\|_2^2\right\} \tag{8}$$

$$\propto \mu(dx) \exp\left\{\langle Y_t, x\rangle - \frac{1}{2t}\|x\|^2\right\}. \tag{9}$$

We see that $\mu_t$ is a random tilt of $\mu$, where $\mu_0 = \mu$ and as $t \to \infty$, $\mu_t \Rightarrow \delta_{x^*}$, "localizing" the random measure. Nonetheless, it is not clear how to use this methodology to sample from $\mu$: one might think of generating the process $\mu_t$, but it depends on $Y_t$ which again depends on $x^* \sim \mu$. However, with the following proposition (proposition

1.1 in [5]) we construct a new diffusion-based algorithm to sample from $\mu$:

**Proposition** Assume $\mu$ has finite second moment. Then, $(Y_t)_{t\geq 0}$ is the unique solution of the following stochastic differential equation (with initial condition $Y_0 = 0$)

$$dY_t = m(Y_t; t)\, dt + dW_t. \tag{10}$$

Here $W_t$ is a standard Brownian motion and

$$m(y; t) := \mathbb{E}[x \mid tx + \sqrt{t}Z = y], \quad x \sim \mu, \quad Z \sim \mathcal{N}(0, I_n). \tag{11}$$

In a nutshell, we discretize the SDE of Eq. (10), for $t \in [0, T]$ for some large $T$, and then use $Y_T/T$ as an approximate sample from $\mu$. (Alternatively, one could also output $m(Y_T; T)$). We also learn the function $m(y, t)$ by training a neural network to predict $x$ given $y$ (by minimizing the empirical risk or other variational losses, like the ELBO).

Even if this method has been constructed using stochastic localization, it is indeed equivalent (up to a change of variables) to the techniques explained in the previous section (5) when approximating the score with Tweedie's Formula. In particular, defining $y = tx + \sqrt{t}Z$, the score of the observation distribution can be expressed as

$$\nabla_y \log \mu_t(y) \approx m(y; t) - \frac{y}{t}. \tag{12}$$

See sections 1.2 and 1.3 of [5] for more details on this equivalence.

# 7 Empirical study on a 2-modes GMM

In this section we start with the example presented in Section 5 of [5]: our goal is to sample from a mixture of two Gaussians with stochastic localization (as we presented in 6); in particular, we aim at understanding how in high dimensions the neural network approximates the score and which architecture is best suited for this task.

We consider a mixture of two well-separated Gaussians in $\mathbb{R}^n$, with centers $a_1, a_2 \in \mathbb{R}^n$, equal covariance matrices $I_n$, and mixture weights $p_1 = p$, $p_2 = 1 - p$. Without loss of generality, we assume that the overall mean $p_1 a_1 + p_2 a_2$ is zero, which allows us to define the model as:

$$\mu = p \cdot \mathcal{N}((1 - p)a; I_n) + (1 - p) \cdot \mathcal{N}(-pa; I_n), \tag{13}$$

where $a := a_1 - a_2$. In our experiments, we follow [5] and fix the parameters as $a = \mathbf{1}$, $p = 0.7$, and $n = 128$. The vector $a$ is chosen so that its norm is proportional to $\sqrt{n}$, ensuring the two components are well separated. In Figure 1 we show an example of samples from $\mu$ and the true PDF projected along the direction of $a$.

Even if this is a toy model, we expect the task of learning the conditional mean/score function to be difficult because of the curse of dimensionality. As the ambient dimension $n$ increases, the volume of the space grows exponentially and the probability mass of the Gaussian components becomes increasingly concentrated in thin regions at radius $\sqrt{n}$. In our setup, for example, although the distribution is defined over $\mathbb{R}^n$, its variability is

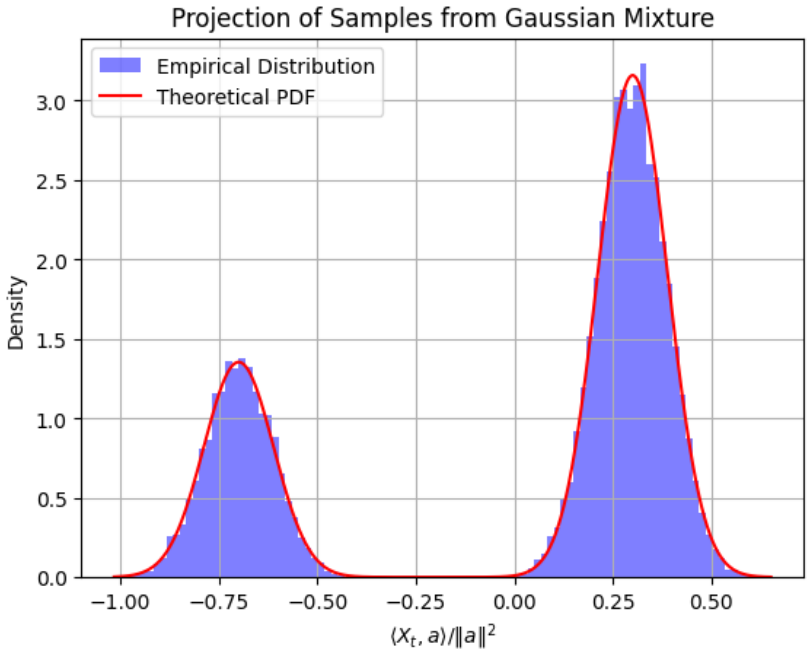


Figure 1: **Samples from $\mu$ we use to train our neural network** The samples are obtained using standard techniques for Gaussians and are projected along the direction $a$ and then rescaled by $||a||^2 = n$. The red line is the true (projected) density function.

essentially confined to the one-dimensional subspace spanned by the mixture separation vector $a$, while the score remains nearly constant in the orthogonal directions. This suggests that the underlying structure is effectively low-dimensional (it will become clear later when we derive it mathematically).

## 7.1 Training and sampling routine

To approximate the conditional expectation $m(y, t)$ from Eq. 11, we train a neural network to recover clean samples $x \sim \mu$ from noisy observations of the form $y_t = tx + \sqrt{t}G$, where $G \sim \mathcal{N}(0, I_n)$. We reparametrize time using an angular variable $\alpha \in [0, \pi/2)$ such that $t = \tan^2(\alpha)$. This transformation maps the unbounded time domain $t \in [0, \infty)$ onto a compact interval, which is beneficial both numerically and theoretically: since neural networks are universal approximators on compact domains, this guarantees their ability to uniformly approximate the dependence of the posterior mean on time.

During training, for each input $x$, a value of $\alpha$ is sampled uniformly from $[0, \alpha_{\max}]$, and the corresponding noisy sample is constructed as $y_t = tx + \sqrt{t}G$. The network receives $y_t$ and $\alpha$ as inputs and is trained to minimize the standard $L^2$ loss between its output and the ground-truth $x$.

This training procedure aligns closely with the loss function proposed in Appendix A of [5], which is derived from minimizing the KL divergence between the target diffusion

process and a learned parametric process. In particular, the empirical loss is

$$\hat{R}_n(\theta) = \frac{1}{n}\sum_{i=1}^{n}\int_0^T \left\|m(Y_t^i) - \hat{m}_\theta(Y_t^i)\right\|^2 \,\mathrm{d}t, \tag{A.5}$$

where $m$ is the optimal drift and $\hat{m}_\theta$ is a learned parametric approximation. In our implementation, the time integral is approximated via Monte Carlo sampling: a single time $t \in [0, T]$ is drawn per sample using the reparametrization $t = \tan^2(\alpha)$, and the network is trained to minimize the squared error

$$\widehat{R}_n(\theta) \approx \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{t,G}\left[\left\|x_i - \hat{m}_\theta(tx_i + \sqrt{t}G; t)\right\|^2\right],$$

where the expectation is over both the noise and the sampled time. This formulation allows the neural network to learn a time-dependent approximation of the optimal drift, effectively minimizing a proxy of the KL divergence between the true diffusion dynamics and the learned generative process.

Now, assuming that the conditional expectation $m(y, t)$ is known or accurately approximated by a neural network denoiser—we generate samples by discretizing the diffusion process defined in Eq. 10 via the Euler–Maruyama method. Again, instead of discretizing time $t$ directly, we construct a uniform grid over the angular variable $\alpha \in [0, \alpha_{\max}]$, and compute the corresponding time steps via the change of variables $t = \tan^2(\alpha)$. At each iteration, the drift is evaluated at the current value of $\alpha$, and the state is updated using the standard Euler–Maruyama update rule. Final samples are normalized by a factor $1/T$, consistent with the forward process. We do not analyze the performance of this discretization scheme in detail, as it is well studied in the literature.

## 7.2 A first attempt using a one layer RELU network

We begin our investigation by implementing the neural network architecture proposed in [5] and analyzing its performance. Montanari chooses to use the RELU as activation function, a choice that at the moment we are mimicking. However, we introduce a slight modification: the skip connection is scaled by a factor of $\cos^2(\alpha)$. The motivation for this change will be provided in the following section.

**Algorithm 1** Forward pass of `TwoLayerDenoiser`

**function** FORWARD($x \in \mathbb{R}^n$, $\alpha \in [0, \pi/2)$)
  $i \leftarrow \{0, 1, \ldots, 20\}$
  $\phi \leftarrow [\cos(\alpha \cdot i),\ \sin(\alpha \cdot i)] \in \mathbb{R}^{42}$ ▷ Fourier time embedding
  $s \leftarrow \mathrm{Linear}_0(\phi) \in \mathbb{R}^L$ ▷ Projected time embedding
  $x_1 \leftarrow \mathrm{ReLU}(\mathrm{Linear}_1(x)) \in \mathbb{R}^{\text{hidden\_dim}}$ ▷ Input projection
  $x_2 \leftarrow$ flatten(outer product of $s$ and $x_1$)
  $x_{\text{out}} \leftarrow \mathrm{Linear}_2(x_2) + \cos^2(\alpha) \cdot x$ ▷ Residual connection
  **return** $x_{\text{out}}$
**end function**

To get an idea of the performance of this network, we trained the denoiser on N samples from the Gaussian mixture distribution $\mu$, and evaluated the quality of the generated samples by projecting them onto the signal direction $a$ and comparing the resulting empirical density with the theoretical (projected) pdf. The following training configurations were tested (top-left to bottom-right in figure 2):

1. $N = 5000$, 500 epochs, $L = 3$, $m = 256$
2. $N = 20000$, 500 epochs, $L = 3$, $m = 256$
3. $N = 20000$, 2000 epochs, $L = 3$, $m = 256$
4. $N = 20000$, 2000 epochs, $L = 6$, $m = 512$

Despite varying the training data size, number of training epochs, and network capacity, none of the configurations accurately match the theoretical distribution. While increasing $N$ from 5000 to 20000 and training for more epochs improves the visual alignment in some regions (near the main mode at $(1-p)$) important differences remain.

These results suggest that the learned posterior mean guides the diffusion process predominantly towards the more probable mode of the target distribution. In other words, the diffusion appears to concentrate near the dominant mode, failing to fully reconstruct the multimodal structure.

This phenomenon motivates the analytical study of the conditional expectation. Since it minimizes the mean squared error in the reconstruction task, understanding its structure might provide key insights into the limits and capabilities of this sampling technique.

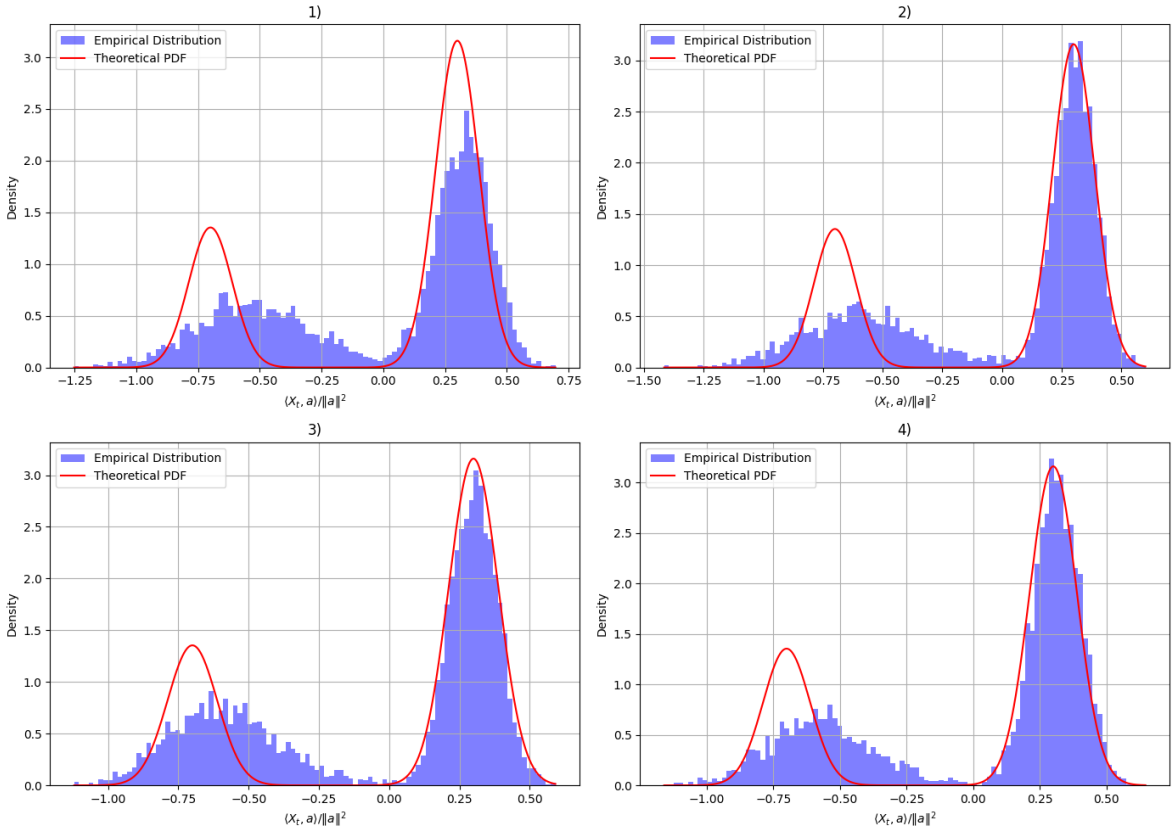


Figure 2: **Projected forward samples under different training configurations.** Each subplot shows the empirical distribution of samples obtained via forward diffusion under different training configurations, as described in the bullet points above.

## 7.3 Mathematical derivation and study of the conditional mean

Recalling that we consider the process:

$$Y_t = tX + \sqrt{t} \cdot G, \quad \text{with } G \sim \mathcal{N}(0, I),$$

where $X \sim p\mathcal{N}(\mu_1, I) + (1-p)\mathcal{N}(\mu_2, I)$, and component means defined as $\mu_1 := (1-p)a$ and $\mu_2 := -pa$, for some fixed $a \in \mathbb{R}^d$. We now look for a closed-form expression for the conditional expectation $m(y,t) := \mathbb{E}[X \mid Y_t = y]$.

Let $i \in \{1, 2\}$ and $Z = i$ if $X$ is generated from the $i$-th Gaussian component. Then, using the law of total expectation:

$$\mathbb{E}(X \mid Y_t) = \mathbb{E}(X \mid Y_t, Z = 1)\,\mathbb{P}(Z = 1 \mid Y_t) + \mathbb{E}(X \mid Y_t, Z = 2)\,\mathbb{P}(Z = 2 \mid Y_t)$$

To compute the term $\mathbb{E}[X \mid Y_t, Z = i]$, we observe that conditioning on $Z = i$ reduces the problem to the case where $X \sim \mathcal{N}(\mu_i, I)$, i.e., a single non-centered Gaussian. In this case, the marginal distribution of $Y_t \mid Z = i \sim \mathcal{N}(t\mu_i, t(t+1)I)$, and since the joint distribution of $(X, Y_t) \mid Z = i$ is Gaussian, the conditional expectation takes the standard form:

$$\begin{aligned} \mathbb{E}[X \mid Y_t, Z = i] &= \mu_i + \mathrm{Cov}(X, Y_t) \cdot \mathrm{Var}(Y_t)^{-1}(Y_t - \mathbb{E}[Y_t]) \\ &= \mu_i + \frac{1}{t+1}(Y_t - t\mu_i) \\ &= \frac{1}{t+1}(Y_t + \mu_i). \end{aligned} \tag{14}$$

This result confirms that, conditioned on the component $Z = i$, the posterior mean is affine in $Y_t$, and reflects the contribution of the corresponding Gaussian component. If we were to design network to sample (using stochastic localization) from a simple non-centered gaussian it would suffice to implement a similar student network and estimate only the (n-dimensional)-parameter $\mu_i$.

Now, let us denote $\gamma_i(Y_t) := \mathbb{P}(Z = i \mid Y_t)$. Using Bayes' rule:

$$\begin{aligned} \mathbb{P}(Z = i \mid Y_t) &= \frac{\mathbb{P}(Y_t \mid Z = i)\,\mathbb{P}(Z = i)}{\mathbb{P}(Y_t)} \\ &= \frac{\mathcal{N}(Y_t; t\mu_i, t(t+1)I) \cdot p_i}{\sum_{j=1}^{2} p_j\,\mathcal{N}(Y_t; t\mu_j, t(t+1)I)} \end{aligned}$$

Therefore the posterior weights are:

$$\gamma_1(y_t) = \frac{p \cdot \exp\left(-\frac{1}{2t(t+1)}\left\|y_t - t(1-p)a\right\|^2\right)}{p \cdot \exp\left(-\frac{1}{2t(t+1)}\left\|y_t - t(1-p)a\right\|^2\right) + (1-p) \cdot \exp\left(-\frac{1}{2t(t+1)}\left\|y_t + tpa\right\|^2\right)} \tag{15}$$

and $\gamma_2(y_t) = 1 - \gamma_1(y_t)$.

An alternative formulation for the posterior weight is using tanh (which follows from the known connection between sigmoid and tanh, see Appendix B for derivation):

$$\gamma_1(y_t) = \frac{1}{2}\left(1 + \tanh\left(\frac{1}{4t(t+1)}\left(\left\|y_t + tpa\right\|^2 - \left\|y_t - t(1-p)a\right\|^2\right) + \frac{1}{2}\log\left(\frac{p}{1-p}\right)\right)\right) \tag{16}$$

This reformulation in terms of the tanh function suggests that tanh activations may be more suitable than ReLU for modeling time dependence in the neural network. We will return to this intuition in the following sections.

In the end, the posterior mean becomes:

$$\mathbb{E}[X \mid Y_t = y_t] = \frac{1}{t+1} y_t + \frac{1}{t+1} \left(\gamma_1(y_t)(1-p)a - \gamma_2(y_t)pa\right)$$

Using $\gamma_2 = 1 - \gamma_1$, this simplifies to the final expression:

$$\mathbb{E}[X \mid Y_t = y_t] = \frac{1}{t+1} y_t + \frac{1}{t+1} (\gamma_1(y_t) - p)a.$$

This shows that the posterior mean lies along the line spanned by $y_t$ and $a$, and that the posterior confidence $\gamma_1(y_t)$ determines the bias toward component 1. When $\gamma_1(y_t) = p$, the posterior mean equals the unconditional mean $\mathbb{E}[X] = 0$.

It also clear that the dependence on the noisy input $y_t$ is simply modulated by a scalar factor $\frac{1}{t+1}$. Since we reparametrize time using $t = \tan^2(\alpha)$, this coefficient becomes

$$\frac{1}{t+1} = \frac{1}{\tan^2(\alpha) + 1} = \cos^2(\alpha).$$

Motivated by this, in the previous paragraph we introduced a time-dependent skip connection of the form $\cos^2(\alpha) \cdot x$. This helps the model better approximate the known structure of the posterior mean in the Gaussian case, but is expected to be beneficial also in more general settings where the posterior mean decomposes similarly (we still have to "guess" the scalar to multiply in front of the skip connection).

Now that we have a closed-form expression for the posterior mean $\mathbb{E}[X \mid Y_t = y_t]$, we can directly compare the output of our neural network to the ground truth. In particular, we project both the predicted and true posterior means onto the direction $a$ and visualize them as functions of $\langle y_t, a \rangle / t$, which corresponds to the natural signal coordinate in this mixture model.

Figure 5 shows this comparison across several values of time $t$. Each subplot reports both the ground truth (blue) and the neural network output (red), along with the relative mean squared error (MSE). As expected, for very small values of $t$, the posterior mean is highly nonlinear and close to a sigmoid in $\langle y_t, a \rangle$, which makes the estimation task more challenging. In these regimes, the network exhibits visible deviations from the ground truth and the MSE is relatively high. An interesting feature emerges in the first two plots, where the neural network output exhibits a sharp right-angle transition along the projection axis. This is probaly a consequence of the use of the ReLU activation function in the first layer which introduces angular discontinuities in the function being approximated.

As $t$ increases, the posterior mean becomes increasingly linear in $y_t$, and the network predictions align more closely with the ground truth. In the large-$t$ regime, the network effectively learns to approximate the identity map $X \approx Y_t/t$, and the approximation error becomes negligible.

Interestingly, while the approximation improves for intermediate values of $t$, we observe a mild increase in the relative error for larger $t$. This behavior is counterintuitive

at first, since the posterior mean becomes nearly linear and the estimation task should, in principle, be easier. However, we hypothesize that this is due to overfitting: in the large-$t$ regime, the noisy input $y_t$ becomes dominated by the signal $tX$, and any residual noise has a diminished effect on the true posterior. Nevertheless, the neural network may still attempt to fit these small fluctuations in the data.

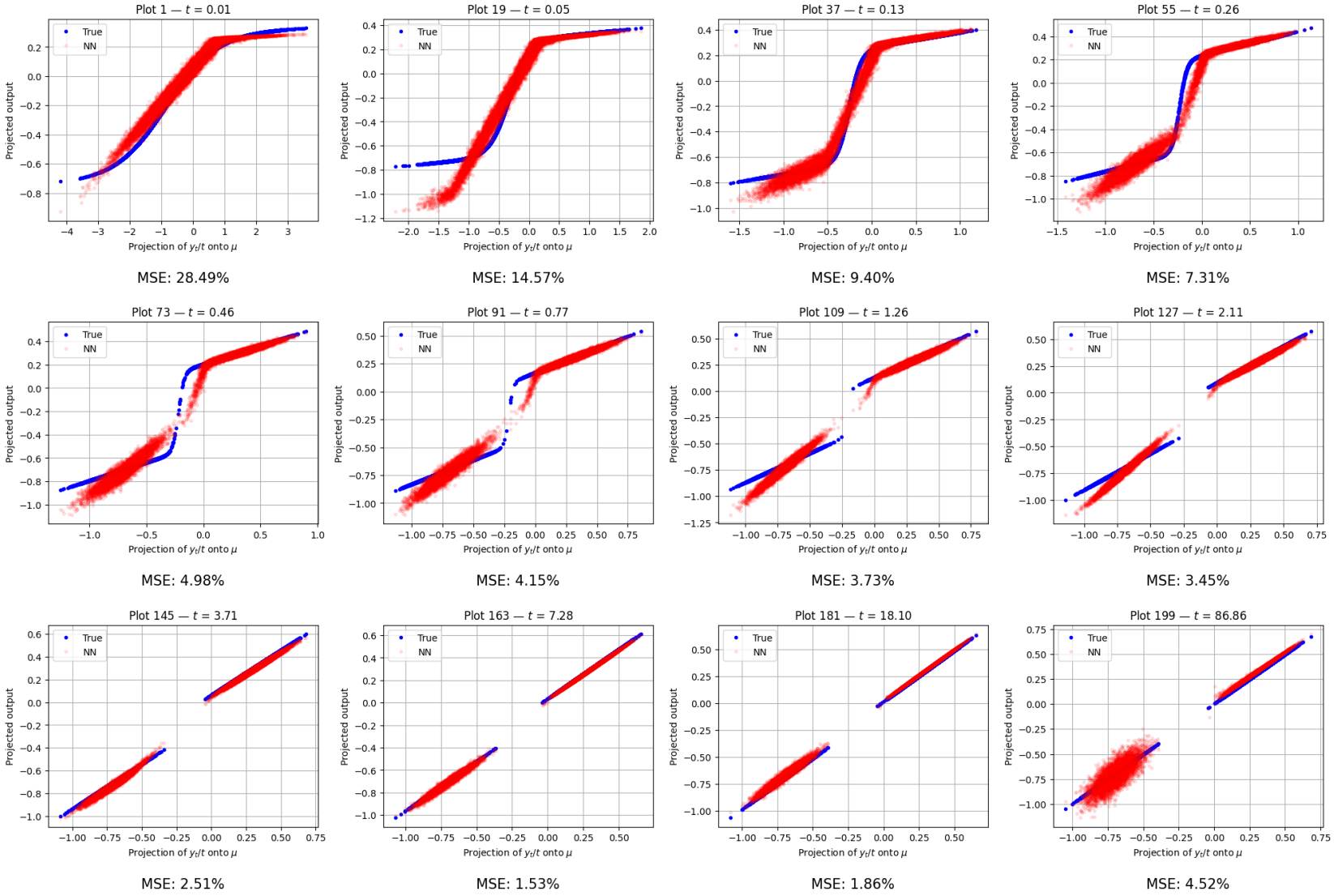


Figure 3: **Comparison between true posterior mean and neural network prediction across different values of $t$ using RELU activation function**

## 7.4 Attempt using a one-layer network with tanh activation

Motivated by the alternative formulation of the posterior weight $\gamma_1(y_t)$ using the tanh function (see Eq. 16), we repeat the experiments described in Section 7.2, modifying only the activation function at line 5 of Algorithm 1, where we replace ReLU with tanh.

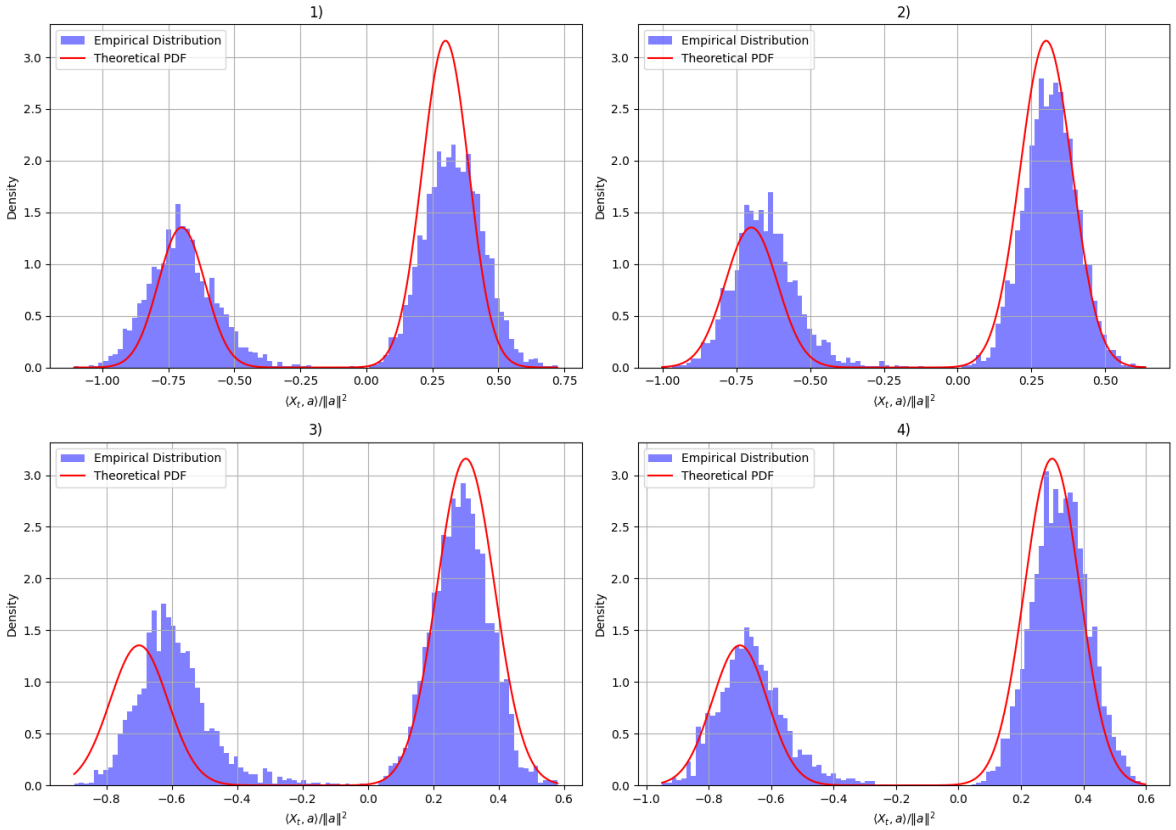


Figure 4: **Projected forward samples under different training configurations (with tanh-network).** Each subplot shows the empirical distribution of samples under the 4 different training configurations and using tanh as activation function.

Visual inspection of Figure 4 suggests that using the tanh activation generally leads to a better visual alignment between the empirical distribution and the theoretical target compared to the ReLU-based network. However, this improvement comes at the cost of slightly reduced precision in sampling around the mode associated with the higher weight $p$, where the generated samples are skewed towards right. In Figure **??**, we again compare the neural network approximation of the conditional expectation against the true function. Although the visual alignment appears generally improved when using tanh, the mean squared error (MSE) at small times is actually larger. This increase in error is due to a poorer approximation around the dominant mode, where approximately 70% of the data is concentrated. This data imbalance was introduced by perturbing the original dataset, which resulted in a higher density of samples near the rightmost mode.

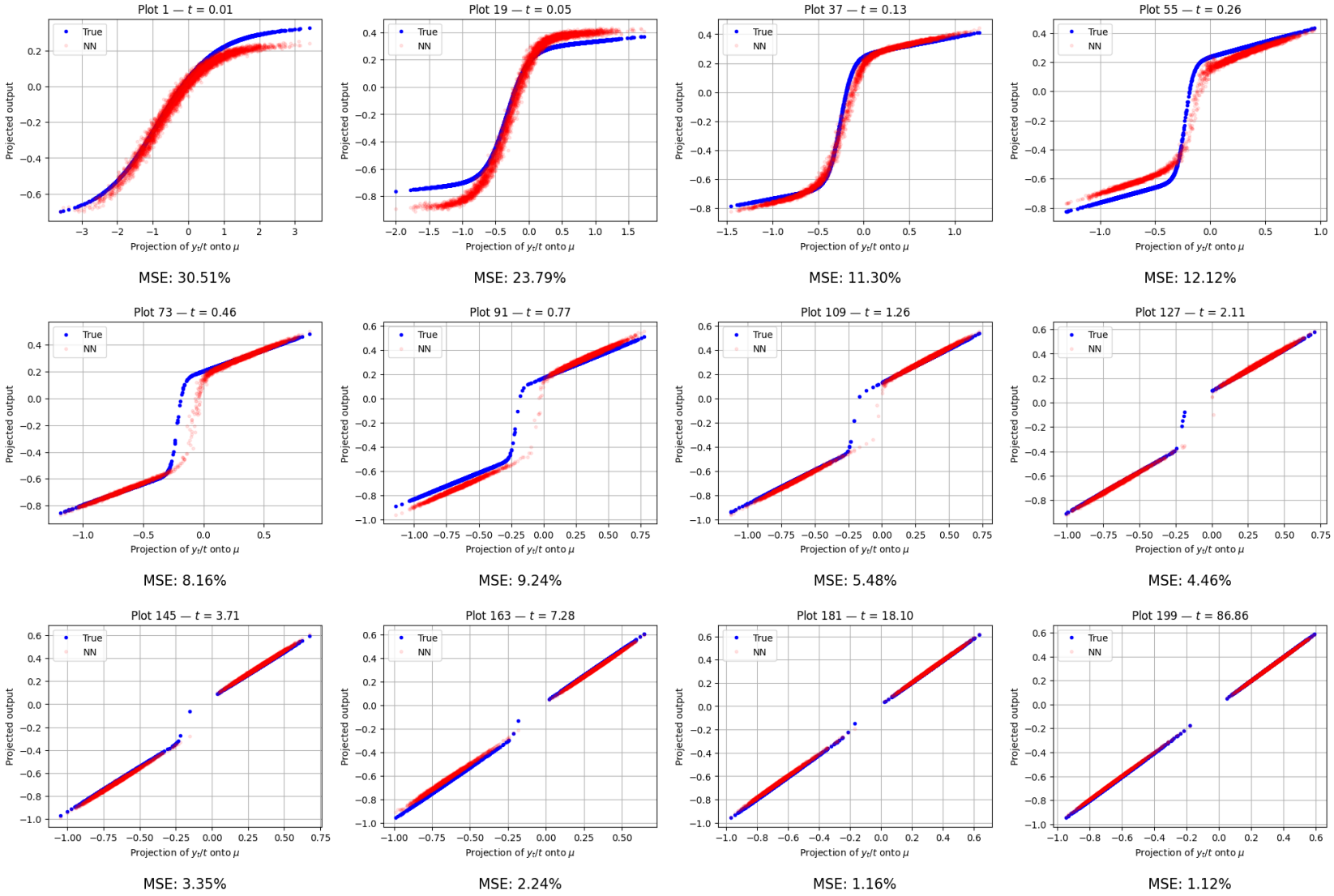


Figure 5: **Comparison between true posterior mean and neural network prediction across different values of $t$ using** tanh **activation function.**

## 7.5 Limiting behaviour of the conditional mean

The mathematical derivation, together with the empirical plots of the conditional mean as $t$ varies, reveals that the posterior exhibits qualitatively different behaviors across different time regimes. This observation naturally motivates an asymptotic analysis to better understand the limiting structures as $t \to 0$ and $t \to \infty$.

**Limit as $t \to 0$.** We have seen that the general form of the posterior mean is:

$$\mathbb{E}[X \mid Y_t = y_t] = \frac{1}{t+1} y_t + \frac{1}{t+1} \left(\gamma_1(y_t) - p\right) a.$$

As $t \to 0$, we have $Y_t \approx \sqrt{t}G$, and hence the first term vanishes:

$$\frac{1}{t+1} y_t = \mathcal{O}(\sqrt{t}) \to 0.$$

Thus, the posterior mean becomes:

$$\mathbb{E}[X \mid Y_t = y_t] \approx \frac{1}{t+1} \left(\gamma_1(y_t) - p\right) a.$$

To approximate $\gamma_1(y_t)$, we expand the likelihood terms using $y_t = \sqrt{t}G$. We compute:

$$\|y_t - t(1-p)a\|^2 = t\|G\|^2 - 2t^{3/2}(1-p)\langle G, a\rangle + t^2(1-p)^2\|a\|^2,$$
$$\|y_t + tpa\|^2 = t\|G\|^2 + 2t^{3/2}p\langle G, a\rangle + t^2p^2\|a\|^2.$$

Substituting into the posterior weight expression 15: and neglecting common terms and lower-order corrections, we obtain:

$$\gamma_1(y_t) \approx \frac{p \cdot \exp\left(\frac{(1-p)}{\sqrt{t}}\langle G, a\rangle\right)}{p \cdot \exp\left(\frac{(1-p)}{\sqrt{t}}\langle G, a\rangle\right) + (1-p) \cdot \exp\left(-\frac{p}{\sqrt{t}}\langle G, a\rangle\right)}.$$

Hence, the posterior mean becomes:

$$\mathbb{E}[X \mid Y_t = y_t] \approx \frac{1}{t+1} \left(\gamma_1(y_t) - p\right) a, \quad \text{with}$$

$$\gamma_1(y_t) \approx \frac{p \cdot \exp\left(\frac{(1-p)}{\sqrt{t}}\langle G, a\rangle\right)}{p \cdot \exp\left(\frac{(1-p)}{\sqrt{t}}\langle G, a\rangle\right) + (1-p) \cdot \exp\left(-\frac{p}{\sqrt{t}}\langle G, a\rangle\right)}.$$

This is approximately a sigmoid in the scalar projection $\langle G, a\rangle$, showing that the posterior mean is aligned with the direction of $a$, and smoothly interpolates based on the noise realization $G$.

**Limit as $t \to \infty$.** As $t \to \infty$, the signal term $tX$ dominates over the noise term $\sqrt{t}G$, so that:

$$Y_t \approx tX \quad \Rightarrow \quad X \approx \frac{Y_t}{t}.$$

The posterior mean is given by:

$$\mathbb{E}[X \mid Y_t = y_t] = \frac{1}{t+1} y_t + \frac{1}{t+1} \left(\gamma_1(y_t) - p\right) a.$$

- The first term satisfies:

$$\frac{1}{t+1} y_t = \frac{t}{t+1} X + \frac{\sqrt{t}}{t+1} G \to X,$$

since $\frac{t}{t+1} \to 1$ and $\frac{\sqrt{t}}{t+1} \to 0$.

- The second term vanishes:

$$\frac{1}{t+1} \left(\gamma_1(y_t) - p\right) a \to 0,$$

because $\gamma_1(y_t) \in [0,1]$ and $\frac{1}{t+1} \to 0$.

Furthermore, the posterior weight $\gamma_1(y_t)$ becomes sharply concentrated. Recall that:

$$\gamma_1(y_t) = \frac{p \cdot \exp\left(-\frac{1}{2t(t+1)} \|y_t - t\mu_1\|^2\right)}{p \cdot \exp(...) + (1-p) \cdot \exp\left(-\frac{1}{2t(t+1)} \|y_t - t\mu_2\|^2\right)}.$$

This expression becomes asymptotically a step function:

$$\gamma_1(y_t) \to \mathbf{1}_{\|\mu_1 - \frac{y_t}{t}\|^2 < \|\mu_2 - \frac{y_t}{t}\|^2},$$

which effectively selects the closest mixture component to $\frac{y_t}{t}$.

However, this sharp decision does not affect the posterior mean in the limit, due to the vanishing prefactor $1/(t+1)$. Hence, we conclude:

$$\mathbb{E}[X \mid Y_t = y_t] \xrightarrow[t \to \infty]{} \frac{y_t}{t}$$

**Intermediate regime and emergence of step-like behavior.** For intermediate values of $t$, we observe a smooth but significant transition between the two limiting behaviors identified in the small and large $t$ regimes. In particular, the posterior mean $\mathbb{E}[X \mid Y_t = y_t]$ exhibits a combination of a linearly growing term and a nonlinear sigmoidal correction. Specifically, the posterior weight $\gamma_1(y_t)$, which controls the bias toward one of the mixture components, progressively sharpens as $t$ increases. For small $t$, this function behaves like a smooth sigmoid in the direction of the mixture separation vector $a$, due to the dominance of the Gaussian noise. Conversely, in the large-$t$ limit, where the forward process becomes almost deterministic ($Y_t \approx tX$), the posterior mean

converges to the linear estimator $\mathbb{E}[X \mid Y_t = y_t] \to y_t/t$, and the nonlinear correction vanishes.

Instead, in the intermediate regime, particularly for values $t \sim \mathcal{O}(1)$, the posterior mean effectively behaves as the superposition of a linear estimator and a sharply transitioning nonlinear component. The function $\gamma_1(y_t)$ becomes increasingly steep and can be approximated as

$$\gamma_1(y_t) \approx \sigma\left(\alpha(t) \cdot \langle y_t, a \rangle\right),$$

where the slope parameter $\alpha(t)$ increases with $t$. As $\alpha(t) \to \infty$, the sigmoid converges pointwise to a Heaviside step function. We define a critical time $t_{\text{sharp}}$ as the smallest value of $t$ such that the derivative of $\gamma_1(y_t)$ with respect to $\langle y_t, a \rangle$ exceeds a fixed threshold $\epsilon^{-1}$, indicating that the sigmoid has become sufficiently sharp to effectively behave like a step. In our empirical plots, this step-like behavior begins to emerge for $t \gtrsim 0.1$, and is fully developed by $t \approx 1$, where the posterior mean exhibits an almost discontinuous transition aligned with the mixture boundary. For larger $t$, the linear term dominates and the overall posterior mean closely follows the bisector $y_t/t$. This mixture of regimes produces a structure that appears almost piecewise affine with a sharp transition zone, despite being globally smooth.

Consequently, the posterior mean takes the approximate form

$$\mathbb{E}[X \mid Y_t = y_t] \approx \frac{1}{t+1} y_t + \frac{1}{t+1} \begin{cases} (1-p)a & \text{if } \langle y_t, a \rangle > \tau(t), \\ -pa & \text{if } \langle y_t, a \rangle < \tau(t), \end{cases} \tag{17}$$

up to small corrections, where $\tau(t)$ is a threshold depending on $t$ and the mixture parameters.

This investigation suggests that a single network may struggle to simultaneously capture both the sharp sigmoidal transition at small $t$ and the linear behavior at large $t$. As a consequence, two strategies naturally emerge:

- **Mixture-aware modeling:** 17 suggests using two separate networks, each specialized in modeling one of the components of the mixture.
- **Phase-aware modeling:** the existence of $t_{sharp}$ suggests instead to split the temporal domain into two regimes and train one network to capture the nonlinear sigmoidal behavior and another to approximate the nearly linear regime.

## 7.6 Attempt with two denoisers (mixture-aware)

As we just said, 17 suggests using two separate networks, each specialized in modeling one of the components of the mixture. We therefore need to split the dataset in two parts with the goal of "assigning" each sample to a mode and train two different denoisers on each subset.

Following the approach suggested in [5], we perform this assignment by projecting the data onto the principal eigenvector $v$ of the empirical covariance matrix $\Sigma = \frac{1}{n}\sum_{i=1} x_i x_i^T$. This is consistent with the interpretation provided by PCA, which identifies the directions of highest variability in the data with the very eigenvectors of the empirical covariance matrix. This direction, since it captures the axis of maximal variance in the data, typically aligns with the direction of separation between the two modes.

In practice, we train a denoiser $m_+(y_t, t)$ on the subset of training points $x$ such that $\langle x, v\rangle > 0$, and a denoiser $m_-(y_t, t)$ on the subset where $\langle x, v\rangle < 0$. During forward sampling, we assign each sample to $m_+$ (i.e we choose which mode we assign the sample we are creating) with probability $q := \frac{|\{x \in \text{dataset} | \langle x, v\rangle > 0\}|}{N}$, and use $m_-$ otherwise. In particular, the assignment of each sample to one of the two denoisers effectively discretizes the posterior weight $\gamma_1(y_t) \in [0, 1]$ into a binary decision $\gamma_1 \in \{0, 1\}$, consistent with the fact that the posterior distribution becomes sharply concentrated around one component. This strategy can be interpreted as using a crude approximation of $\gamma_1(y_t)$, replacing its smooth sigmoidal transition with a hard threshold.

We then follow the same sampling procedure described in Section 7.1. If the dataset is sufficiently large and well balanced, we expect $q \approx p$ (where $p$ is the true weight of the first mode in the mixture).

We repeated the same experiment in the four settings described in Section 7.2. using both tanh and RELU. In Figures 6, we report the results. Comparing these results with the previous ones (Figures 2 and 4), we observe a significant improvement in the quality of the sampling: the empirical distributions now align much more closely with the theoretical targets, especially around the modes. Using a separate denoiser per mode effectively removes the approximation issues we previously observed near the dominant mode, leading to both better visual alignment and lower sampling errors. The difference between tanh and relu activations is now less pronounced compared to the single-denoiser setting, as the both networks are capable of approximating the simplified (unimodal) posterior mean.

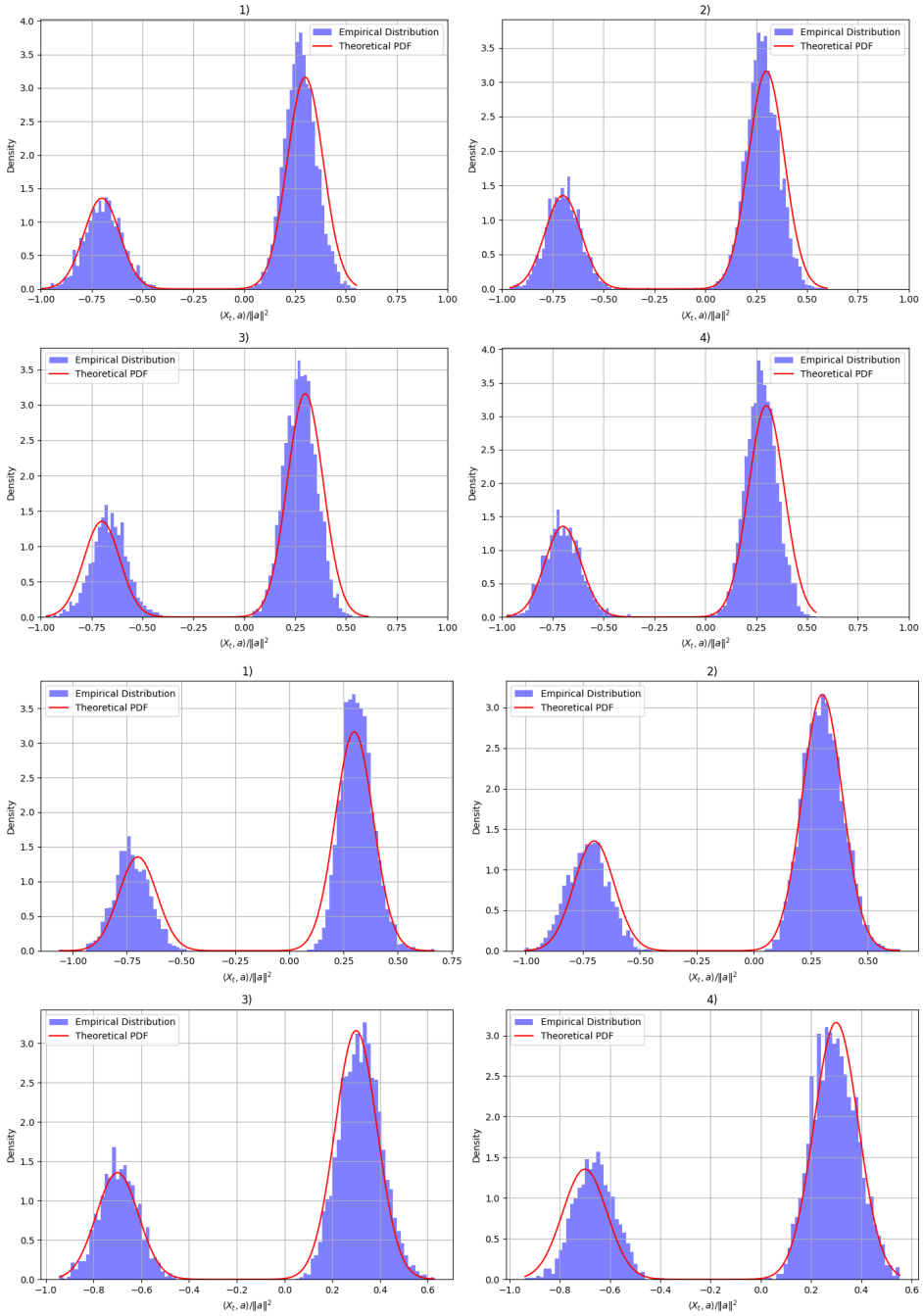


Figure 6: **Projected forward samples under different training configurations using two denoisers, one per mode.** Top: results using relu activation; Bottom: results using tanh activation. Each subplot shows the empirical distribution of samples under four different training configurations. Compared to the single-denoiser setting, a substantial improvement in the match with the theoretical distribution is observed, particularly around the modes.

## 7.7 Attempt with two denoisers (phase-aware modeling)

Motivated by the reasoning at the end of section 7.5, we train two separate networks (with identical architectures) specialized for the two different regimes (but a direction of study could be to use specific architecture for each regime): a *sigmoidal regime* for $t \in [0, t_{\text{sharp}}]$, and a *linear regime* for $t > t_{\text{sharp}}$. We choose $t_{\text{sharp}} = 1$, as discussed in Section 7.5, since this is the point where the step-like behavior begins to emerge in our empirical plots. During sampling, we simply switch the network used to compute the drift of the SDE once $t$ exceeds $t_{\text{sharp}}$. We then repeat the same experiment in the four settings described in Section 7.2 using tanh.

Now, by looking at Figure 7, we see that compared to the previous settings (single network, two denoisers, or per-mode denoisers), this configuration achieves the best match between the empirical and theoretical distributions across all training setups. Both modes are now accurately captured, and the transitions between them are handled much more smoothly. Overall, as shown in Figure 8, using separate networks for each regime leads to a significant reduction in the mean squared error compared to a regime-agnostic approach. This confirms that regime-specific specialization significantly enhances the quality of the learned score functions.

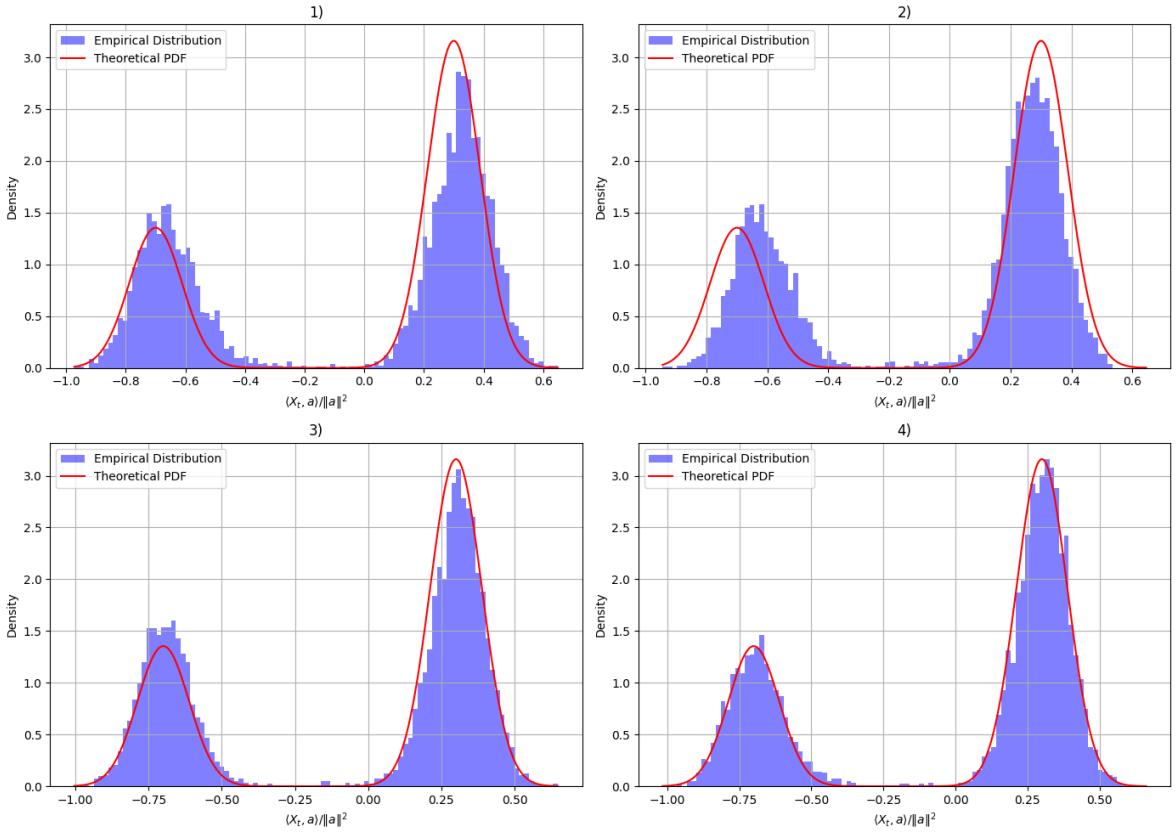


Figure 7: **Projected forward samples under different training configurations using two denoisers, one per regime.** Each subplot shows the empirical distribution of samples obtained via forward diffusion under different training configurations.

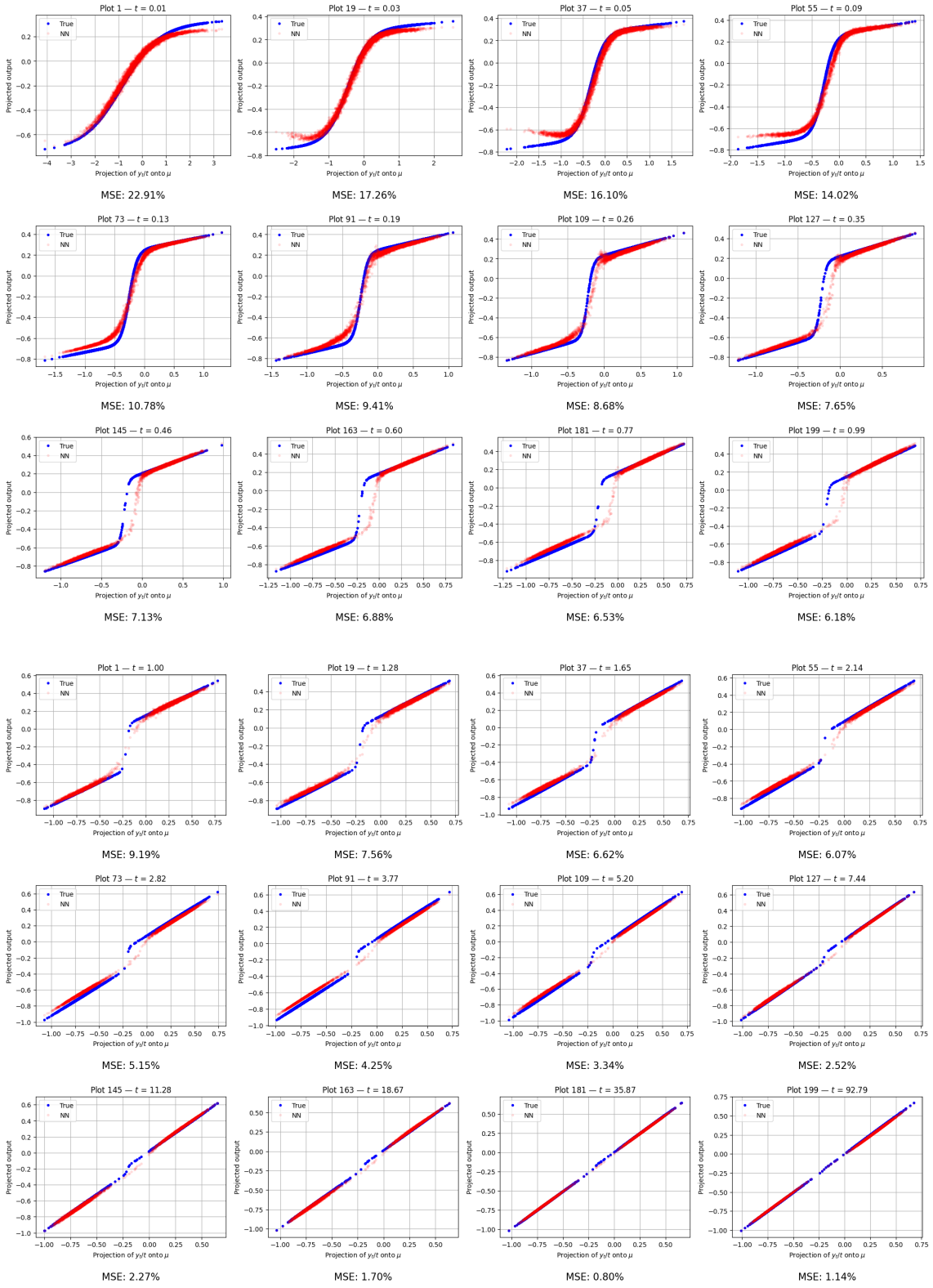


Figure 8: **Comparison between true posterior mean and neural network prediction across different values of $t$.** The top panel shows results for the sinusoidal regime, while the bottom panel corresponds to the linear regime

# 8 Convergence analysis in a toy case

## 8.1 Convergence for the DDPM objective

In this final section of the semester project, we study the convergence of learning a GMM with two modes and constant-separated centers using stochastic localisation. This setting has been thoroughly analyzed by Shah, Chen, and Klivans in [13], specifically in the context of the DDPM objective.

In the first part of their paper, the authors consider the case of equation (13) with $p = \frac{1}{2}$ and $a = 2\mu^*$, which corresponds to the distribution

$$q = \frac{1}{2} \cdot \mathcal{N}(\mu^*; I_n) + \frac{1}{2} \cdot \mathcal{N}(-\mu^*; I_n). \tag{18}$$

Their analysis employs a student network of the form

$$s_{\theta_t}(x) = \tanh(\mu_t^\top x)\, \mu_t - x, \quad \text{where} \quad \mu_t = \mu \exp(-t),$$

with $\mu$ as the sole trainable parameter. This network is the exact minimizer of the DDPM loss

$$\min_{s_t} L_t(s_t) = \mathbb{E}_{X_0, Z_t} \left[ \left\| s_t(X_t) + \frac{Z_t}{\sqrt{1 - \exp(-2t)}} \right\|^2 \right],$$

when $\mu = \mu^*$, as it matches the closed-form expression of the score function in this setting.

In the paper, the authors prove that $\mu^*$ can be recovered through the following two-stage procedure:

- **Stage 1 (High noise regime:** Gradient descent is applied to the DDPM objective (as described in Algorithm 1), starting from a random Gaussian initialization. This step uses a high noise scale $t_1 = \mathcal{O}(\log d)$, and runs for a fixed number of iterations $H = \text{poly}(d, 1/\varepsilon)$.
- **Stage 2 (Low noise regime:** The output of the first stage is used as initialization. The algorithm is then run again on the DDPM objective at a low noise scale $t_2 = \mathcal{O}(1)$.

They prove the convergence of this procedure by leveraging connections with well-known algorithms in two distinct regimes:

- **High noise regime:** When the noise level is large, gradient descent on the DDPM loss approximates a single step of *power iteration* on the matrix

  $$\mu - \eta \nabla_\mu L_t(s_\mu) \approx \left( (1 - \eta r)\, \mathrm{Id} + 2\eta \mu_t^* \mu_t^{*\top} \right) \mu$$

  where $\mu_t^* = \mu \exp(-t)$ is the top eigenvector (aligned with the true mean direction), and $r$ is a scalar depending on $\mu$. Starting from a random Gaussian initialization, repeated gradient updates reduce the angular error and allow convergence to $\mu_t^*$. This justifies the use of large $t_1$ in the first stage of the algorithm. After this first stage the parameter $\mu$ is aligned with $\mu^*$ but differs in norm.

- **Low noise regime ($t \ll 1$):** When the noise is small, the DDPM gradient closely approximates the *M-step* in the Expectation-Maximization (EM) algorithm for learning mixtures of Gaussians. Specifically, the update takes the form

$$\mu - \eta \nabla_\mu L_t(s_\mu) \approx (1-\eta)\mu + \eta\, \mathbb{E}_{X \sim \mathcal{N}(\mu_t^*; I_n)}[\tanh(\langle \mu, X \rangle) X]$$

  which matches the update in EM for symmetric two-component Gaussians. Under the assumption that $\mu$ is sufficiently aligned with the ground truth $\mu^*$, convergence to $\mu^*$ is guaranteed. This motivates running the second stage of the algorithm at a small noise level $t_2 = \mathcal{O}(1)$, initialized from the high-noise solution. Eventually this second part simply rescales the parameter $\mu$.

In the paper, the authors also extend their analysis to settings with small separation between components and to mixtures with a general number K of modes, showing convergence guarantees in both cases; however, we do not explore these extensions in this work.

## 8.2 Convergence analysis for Stochastic localization

We now try to mimic the methodology of the previous section (without providing convergence guarantees) for the stochastic localization loss:

$$\mathbb{E}_{X_0, G}\left[\left\| X_0 - \hat{m}_\theta(tX_0 + \sqrt{t}G; t)\right\|^2\right],$$

using the following student network (with again $\mu$ as the sole trainable parameter):

$$\hat{m}_\theta(y;t) = \frac{1}{t+1}\left(y + \tanh\left(\frac{\langle y, \mu\rangle}{t+1}\right)\mu\right).$$

This form corresponds to the conditional mean derived in Section 7.3 for $p = \frac{1}{2}$ (see Appendix C for full derivation), and it is the true minimizer of the loss when $\mu = \mu^*$.

As discussed in Section 6, stochastic localization with Gaussian noise is equivalent (up to a change of variables) to the DDPM formulation based on OU process. Therefore, we expect the behavior in this setting to closely mirror that of the previous subsection. To test this intuition we run the two-stage procedure (Stage 1 and Stage 2) with the current loss and current student network. Note that, now, time is effectively reversed: small values of $t$ correspond to the high-noise (low-signal) regime, while large values of $t$ correspond to the low-noise (high-signal) regime.

In this experiment, we set the dimension of the Gaussian mixture model to $d = 128$, and define the true mean as $\mu^* = 20 \cdot \mathbf{1}$. We initialize $\mu$ randomly and run gradient descent in the high-noise regime with $t = 0.1$ for $H = 500$ steps. We observe that the cosine similarity between $\mu$ and $\mu^*$,

$$\cos(\theta) = \frac{\langle \mu, \mu^* \rangle}{\|\mu\|\,\|\mu^*\|},$$

converges to 1 (see Figure 9), indicating that $\mu$ becomes aligned with $\mu^*$. In fact, at the end of this phase, we typically observe that $\mu \approx 10 \cdot \mathbf{1}$, demonstrating that the optimization learns the correct direction but underestimates the magnitude.

In the second stage, we switch to the low-noise regime by setting $t = 5$, and continue the gradient descent for $10^6$ steps to refine the norm of $\mu$. As shown in the second plot, the distance $\|\mu - \mu^*\|$ converges to zero, confirming that this two-stage procedure achieves full recovery of the true mean vector.

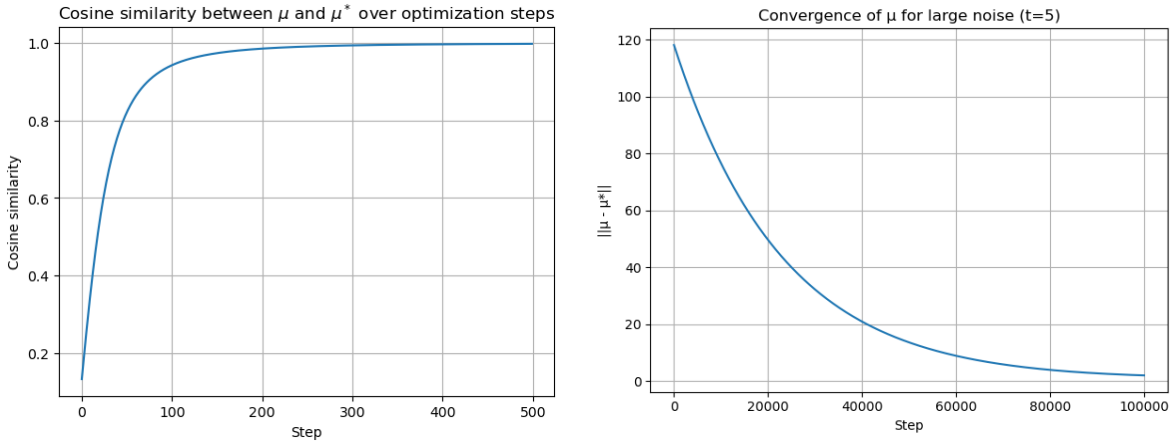


Figure 9: Two-stage optimization: the first stage aligns $\mu$ with $\mu^*$, as shown by the cosine similarity approaching 1. The second stage rescales the norm of $\mu$, driving $\|\mu - \mu^*\|$ to zero.

This section further demonstrates that training a neural network to approximate the conditional mean for forward diffusion (using the stochastic localization objective) naturally unfolds in two phases, provided that the network architecture is appropriately chosen; that is, it resembles the true conditional mean. This observation somehow justifies our approach of *phase-aware modeling*, implemented in Section 7.7

Now we study the behaviour of $\hat{m}_\theta(y; t)$ and the gradient of the loss for small t and big t.

### a. High noise regime

In the regime $t \ll 1$, we note that $a = \frac{1}{1+t} \approx 1$, and that the input to the network becomes is dominated by noise. In this case, the denoiser learned by the network takes the approximate form

$$m_\theta(\mu) \approx y_t + \langle y_t, \mu \rangle \mu.$$

The gradient of the loss becomes approximately

$$\nabla_\mu \, \mathbb{E}_{X_0, G} \left[ \| X_0 - y_t - \langle y_t, \mu \rangle \mu \|^2 \right],$$

This expression reveals that the network adjusts $\mu$ to align with directions in which $y_t$ carries useful information about $X_0$.

This phase can be interpreted as estimating the correct *direction* of the mean, while the correct *scale* is recovered later in the low-noise regime.

For small t we have the following approximation (see Appendix D for full derivation):

$$\mathbb{E}[\nabla_\mu L] \approx 2t \left[ -2(\mu^\top \mu^*)\mu^* + \|\mu\|^2 \mu \right].$$

and therefore a gradient descent step with learning rate $\eta$ gives:

$$\mu \leftarrow \mu - \eta \cdot 2t \left[-2(\mu^\top \mu^*)\mu^* + \|\mu\|^2 \mu\right].$$

This can be written in matrix form as:

$$\mu - \eta\, \mathbb{E}[\nabla_\mu L(\mu)] \approx \left(I - 2\eta t\|\mu\|^2 I + 4\eta t \mu^* (\mu^*)^\top\right) \mu.$$

.

We have derived the following approximate update:

$$\mu^{(k+1)} \approx A^{(k)} \mu^{(k)},$$

which is structurally similar to a matrix with dominant eigenvector $\mu^*$. Indeed, $A^{(k)}$ is symmetric and $\mu^*$ is an eigenvector with eigenvalue: $\lambda_1^{(k)} = 1 - 2\eta t\|\mu^{(k)}\|^2 + 4\eta t\|\mu^*\|^2$ while all orthogonal directions have eigenvalue: $\lambda_0^{(k)} = 1 - 2\eta t\|\mu^{(k)}\|^2$. Since $\|\mu^*\|^2 > 0$, then $\lambda_1^{(k)} > \lambda_0^{(k)}$.

Therefore, the update behaves like a power iteration to extract the direction of $\mu^*$, with an additional term that modulates the norm. This is (as expected) very similar to what is derived in [13] for the low noise regime and validates the first part of our procedure above.

### b. Low noise regime

Remembering that $y_t = tX_0 + \sqrt{t}G$, we approximate:

$$\frac{\langle y_t, \mu\rangle}{t+1} \approx \langle X_0, \mu\rangle.$$

Thus, the denoiser becomes approximately:

$$\frac{1}{t+1}\left(tX_0 + \tanh(\langle X_0, \mu\rangle)\mu\right).$$

And therefore the denoising loss reduces to:

$$\mathbb{E}_{X_0}\left[\left\|X_0 - \frac{1}{t+1}\left(tX_0 + \tanh(\langle X_0, \mu\rangle)\mu\right)\right\|^2\right] = \frac{1}{(t+1)^2}\mathbb{E}_{X_0}\left[\|X_0 - \tanh(\langle X_0, \mu\rangle)\mu\|^2\right]$$

Now, after iterating in the high noise regime, the algorithm managed to align $\mu$ with $\mu^*$ and therefore we can suppose that $\mu = \alpha\mu^*$ for some $\alpha \in \mathbb{R}$. This result and the shape of the above loss function allows us to conclude that gradient descent preserves the alignemnt to $\mu^*$, effectively reducing the problem to a one-dimesional optimization w.r.t $\alpha$. Indeed,

$$\mathcal{L}(\mu) = \frac{1}{(t+1)^2}\mathbb{E}\left[\|X_0 - \alpha\tanh(\alpha\langle X_0, \mu^*\rangle)\mu^*\|^2\right] =: \tilde{\mathcal{L}}(\alpha),$$

i.e., the loss depends only on the scalar $\alpha$, not on the direction of $\mu$ (which is fixed as $\mu^*$) and by the chain rule:

$$\nabla_\mu \mathcal{L}(\mu) = \frac{d\tilde{\mathcal{L}}}{d\alpha} \cdot \nabla_\mu(\alpha\mu^*) = \tilde{\mathcal{L}}'(\alpha) \cdot \mu^*.$$

Therefore, for all $\mu = \alpha\mu^*$, the gradient $\nabla_\mu \mathcal{L}(\mu) \in \text{span}(\mu^*)$. Now, using the standard update:

$$\mu_{t+1} = \mu_t - \eta \nabla_\mu \mathcal{L}(\mu_t),$$

we obtain:

$$\mu_{t+1} = \alpha_t \mu^* - \eta \cdot \tilde{\mathcal{L}}'(\alpha_t) \cdot \mu^* = \alpha_{t+1}\mu^*,$$

which means that at each iterate $\mu_t \in \text{span}(\mu^*)$ for all $t \in \mathbb{N}$.

The problem now is one-dimensional and the loss has the form:

$$\mathcal{L}(\alpha) = \mathbb{E}_{X_0}\left[\|X_0 - \alpha \tanh(\alpha\langle X_0, \mu^*\rangle)\mu^*\|^2\right],$$

which closely resembles the EM update when the denoiser implements the posterior mean.

Then, the loss is minimized at $\alpha = 1$, since this is the value at which the scaling of the projection $\langle X_0, \mu\rangle$ is correct and the denoiser aligns with the Bayes-optimal responsibility function for the GMM.

In conclusion running the two stage procedure using the stochastic localisation loss will effectively recover the true paramter $\mu^*$.

# References

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN. *ICML*, 2017.

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

[3] A. Hyvärinen and P. Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6, 2005.

[4] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. *arXiv preprint arXiv:1312.6114*, 2014.

[5] A. Montanari. Sampling, Diffusions, and Stochastic Localization. *arXiv preprint arXiv:2305.10690*, 2023.

[6] Y. Song and S. Ermon. Generative modeling by estimating gradients of the data distribution. *NeurIPS*, 2019.

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

[8] P. Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7), 2011.

[9] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations (ICLR)*, 2017.

[10] A. van den Oord, O. Vinyals, and K. Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[11] A. Krause and J. Hübottter. Probabilistic Artificial Intelligence.

[12] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 2020.

[13] K. Shah, S. Chen, and A. Klivans. Learning Mixtures of Gaussians Using the DDPM Objective. *arXiv preprint arXiv:2307.01178*, 2023.

[14] Y. Song, S. Garg, J. Shi, and S. Ermon. Sliced Score Matching: A Scalable Approach to Density and Score Estimation. *arXiv preprint arXiv:1905.07088*, 2019.

# Acknowledgments

This project has been conducted with the help of artificial intelligence tools, which have supported technical writing and code debugging/writing. However, all interpretations, mathematical derivations, methodological choices, and conclusions presented in this work remain the sole responsibility of the author.

Furthermore, the first part of this work relates to the lecture notes from Professor Yuansi Chen's course on *Diffusion, Sampling, and Stochastic Localization*, held at the ETH Zurich in Spring 2024.

# A Proof of equivalent formulation for DSM

We aim to derive the following expression:

$$\mathbb{E}_{X\sim\mu,Z\sim\mathcal{N}(0,\sigma^2 I_n)}\left\|s_\theta(X+\sigma Z)+\frac{Z}{\sigma}\right\|_2^2. \tag{19}$$

starting from the standard score matching objective:

$$\mathbb{E}_{X\sim\mu,Z\sim\mathcal{N}(0,I)}\left[\|s_\theta(\tilde{X})\|^2+2\nabla_{\tilde{x}}\cdot s_\theta(\tilde{X})\right]. \tag{20}$$

where $\tilde{X}=X+\sigma Z$.

## A.1 Step 1: Gradient of the Log-Density

The perturbed density $\mu_\sigma$ is defined as:

$$\mu_\sigma(\tilde{x})=\int \mu(x)\mathcal{N}(\tilde{x};x,\sigma^2 I)dx. \tag{21}$$

Taking its log-gradient:

$$\nabla_{\tilde{x}}\log\mu_\sigma(\tilde{x})=\frac{\nabla_{\tilde{x}}\mu_\sigma(\tilde{x})}{\mu_\sigma(\tilde{x})}. \tag{22}$$

Using the property of Gaussian convolution, this can be rewritten as:

$$\nabla_{\tilde{x}}\log\mu_\sigma(\tilde{x})=\mathbb{E}_{X\sim\mu(x|\tilde{x})}\left[\frac{X-\tilde{x}}{\sigma^2}\right]. \tag{23}$$

Applying the reparameterization $\tilde{X}=X+\sigma Z$ with $Z\sim\mathcal{N}(0,I)$, we obtain:

$$\frac{X-\tilde{X}}{\sigma^2}=-\frac{Z}{\sigma}. \tag{24}$$

Thus, we conclude:

$$\nabla_{\tilde{x}}\log\mu_\sigma(\tilde{x})=-\frac{Z}{\sigma}. \tag{25}$$

## A.2 Step 2: Integration by Parts

We now substitute this result into the score matching objective:

$$\mathbb{E}_{X\sim\mu,Z\sim\mathcal{N}(0,I)}\left[\|s_\theta(\tilde{X})\|^2+2\nabla_{\tilde{x}}\cdot s_\theta(\tilde{X})\right]. \tag{26}$$

Using an integration by parts argument, we leverage the identity:

$$\mathbb{E}_{\tilde{X}\sim\mu_\sigma}[\nabla_{\tilde{x}}\cdot s_\theta(\tilde{X})]=-\mathbb{E}_{\tilde{X}\sim\mu_\sigma}\left[\left\langle s_\theta(\tilde{X}),\nabla_{\tilde{x}}\log\mu_\sigma(\tilde{X})\right\rangle\right]. \tag{27}$$

Substituting $\nabla_{\tilde{x}}\log\mu_\sigma(\tilde{x})=-Z/\sigma$, we obtain:

$$\mathbb{E}\left[2\nabla_{\tilde{x}}\cdot s_\theta(\tilde{X})\right]=-2\mathbb{E}\left[\left\langle s_\theta(\tilde{X}),\frac{Z}{\sigma}\right\rangle\right]. \tag{28}$$

### A.3 Step 3: Completing the Square

Rewriting the objective function:

$$\mathbb{E}\left[\|s_\theta(\tilde{X})\|^2 - 2\left\langle s_\theta(\tilde{X}), \frac{Z}{\sigma}\right\rangle\right]. \tag{29}$$

Using the identity:

$$\|a\|^2 - 2\langle a, b\rangle = \|a - b\|^2 - \|b\|^2, \tag{30}$$

we complete the square and obtain:

$$\mathbb{E}\left[\left\|s_\theta(\tilde{X}) + \frac{Z}{\sigma}\right\|_2^2\right], \tag{31}$$

which is exactly the desired equation:

$$\mathbb{E}_{X\sim\mu, Z\sim\mathcal{N}(0,\sigma^2 I_n)}\left\|s_\theta(X + \sigma Z) + \frac{Z}{\sigma}\right\|_2^2. \tag{32}$$

# B Derivation of $\gamma_1(y_t)$ in $\tanh$ form

We start from the original expression for the posterior weight:

$$\gamma_1(y_t) = \frac{p \cdot \exp\left(-\frac{1}{2t(t+1)}\left\|y_t - t(1-p)a\right\|^2\right)}{p \cdot \exp\left(-\frac{1}{2t(t+1)}\left\|y_t - t(1-p)a\right\|^2\right) + (1-p)\cdot \exp\left(-\frac{1}{2t(t+1)}\left\|y_t + tpa\right\|^2\right)}. \tag{33}$$

We rewrite this expression as a logistic sigmoid by factoring out the common exponential term from the numerator and denominator. Define:

$$z := \frac{1}{2t(t+1)}\left(\|y_t + tpa\|^2 - \|y_t - t(1-p)a\|^2\right).$$

Then, we obtain:

$$\gamma_1(y_t) = \frac{1}{1 + \frac{1-p}{p}\exp(-z)} = \sigma\left(z + \log\left(\frac{p}{1-p}\right)\right),$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the logistic sigmoid.

Now, using the identity:

$$\sigma(x) = \frac{1}{2}(1 + \tanh(x/2)),$$

we rewrite $\gamma_1(y_t)$ as:

$$\gamma_1(y_t) = \frac{1}{2}\left(1 + \tanh\left(\frac{1}{4t(t+1)}\left(\|y_t + tpa\|^2 - \|y_t - t(1-p)a\|^2\right) + \frac{1}{2}\log\left(\frac{p}{1-p}\right)\right)\right). \tag{34}$$

# C Derivation of the Conditional Expectation

We report here the full derivation of the conditional expectation $\mathbb{E}(X \mid Y_t = y_t)$ for the mixture of two Gaussians with symmetric means and equal weights, when $p = \frac{1}{2}$. The posterior mean simplifies to:

$$\mathbb{E}(X \mid Y_t = y_t) = \frac{1}{t+1} y_t + \frac{1}{t+1} \left( \delta_1(y_t) - \frac{1}{2} \right) \mu$$

We compute $\delta_1(y_t)$:

$$\begin{aligned} \delta_1(y_t) &= \frac{1}{2} \left( 1 + \tanh \left( \frac{1}{2(t+1)} \left( \|y_t + \frac{1}{2}\mu\|^2 - \|y_t - \frac{1}{2}\mu\|^2 \right) \right) \right) \\ &= \frac{1}{2} \left( 1 + \tanh \left( \frac{1}{t+1} \langle y_t, \mu \rangle \right) \right) \end{aligned}$$

Therefore, we can write:

$$\begin{aligned} \mathbb{E}(X \mid Y_t = y_t) &= \frac{1}{t+1} y_t + \frac{1}{t+1} \left( \frac{1}{2} + \frac{1}{2} \tanh \left( \frac{1}{t+1} \langle y_t, \mu \rangle \right) - \frac{1}{2} \right) \mu \\ &= \frac{1}{t+1} \left( y_t + \tanh \left( \frac{1}{t+1} \langle y_t, \mu \rangle \right) \mu \right) \end{aligned}$$

This matches the student network architecture introduced in the main text.

# D Derivation of approximate gradient in the high noise regime

**Loss function:**

$$\mathcal{L}(\mu) = \|X_0 - y_t - \langle y_t, \mu \rangle \mu\|^2 .$$

**Expansion:**

$$\mathcal{L}(\mu) = \left\| (1-t)X_0 - \sqrt{t}G - \langle tX_0 + \sqrt{t}G, \mu \rangle \mu \right\|^2 .$$

**Gradient (before expectation):**

$$\nabla_\mu \mathcal{L}(\mu) = -2 \left( \langle y_t, \mu \rangle (X_0 - y_t) + \langle X_0 - y_t, \mu \rangle y_t \right) + 2\langle y_t, \mu \rangle^2 \mu + 2\|\mu\|^2 \langle y_t, \mu \rangle y_t.$$

**Moments:**

$$\mathbb{E}[X_0] = 0, \quad \mathbb{E}[X_0 X_0^\top] = I + \mu^* \mu^{*\top},$$
$$\mathbb{E}[X_0 G^\top] = 0 \quad \text{(independence)}, \quad \mathbb{E}[GG^\top] = I.$$

## Term-by-term computation

**1. First term**

$$\mathbb{E}[-\langle \mu, y_t \rangle (X_0 - y_t)] = -\mathbb{E}[\langle \mu, y_t \rangle X_0] + \mathbb{E}[\langle \mu, y_t \rangle y_t]$$

$$\begin{aligned}\mathbb{E}[\langle \mu, y_t \rangle X_0] &= \mu^\top \mathbb{E}[y_t X_0^\top] = \mu^\top (t\mathbb{E}[X_0 X_0^\top]) = t\mu^\top (I + \mu^* \mu^{*\top}) \\ &= t(\mu^\top \mu^*)\mu^* + t\mu,\end{aligned}$$

$$\begin{aligned}\mathbb{E}[\langle \mu, y_t \rangle y_t] &= \mu^\top \mathbb{E}[y_t y_t^\top] = \mu^\top (t^2 (I + \mu^* \mu^{*\top}) + tI) \\ &= t^2 (\mu^\top \mu^*)\mu^* + (t^2 + t)\mu.\end{aligned}$$

So the first term becomes:

$$-t(\mu^\top \mu^*)\mu^* + \mathcal{O}(t^2).$$

**2. Second term**

$$\mathbb{E}[-(X_0 - y_t)^\top \mu \cdot y_t] = -\mathbb{E}[X_0^\top \mu \cdot y_t] + \mathbb{E}[\langle \mu, y_t \rangle y_t]$$

$$\mathbb{E}[X_0^\top \mu \cdot y_t] = t(\mu^\top \mu^*)\mu^* + t\mu \quad \text{(same as above)}.$$

So the second term is:

$$-t(\mu^\top \mu^*)\mu^* + \mathcal{O}(t^2).$$

**3. Third term**

$$\mathbb{E}[(\mu^\top y_t)^2 \mu] = \mathbb{E}[(\mu^\top y_t)^2]\mu$$

$$\mathbb{E}[(\mu^\top y_t)^2] = \mathrm{Var}(\mu^\top y_t) = \mu^\top \mathbb{E}[y_t y_t^\top]\mu = t^2(\mu^\top \mu^*)^2 + (t^2 + t)\|\mu\|^2$$

$$\Rightarrow \mathbb{E}[(\mu^\top y_t)^2 \mu] = t\|\mu\|^2 \mu + \mathcal{O}(t^2).$$

**4. Fourth term**

$$\mathbb{E}[\|\mu\|^2 \langle y_t, \mu \rangle y_t] = \|\mu\|^2 \cdot \mathbb{E}[\langle y_t, \mu \rangle y_t] = \|\mu\|^2 \cdot \left[t(\mu^\top \mu^*)\mu^* + t\mu\right]$$

$$= t\|\mu\|^2 \mu + \mathcal{O}(t^2)$$

**Final result**

$$\mathbb{E}[\nabla_\mu \mathcal{L}(\mu)] = 2t\left[-2(\mu^\top \mu^*)\mu^* + \|\mu\|^2 \mu\right] + \mathcal{O}(t^2)$$