Pittsburgh Interdisciplinary Mathematics Review **4** (2026), 1–17

Diffusion models

A high level mathematical overview

Stephan Wojtowytsch

(Communicated by Lark Song
Copyedited by Bridget Vaughan)

1. Introduction

Diffusion models are tools of deep learning which underpin AI image generators such as Stable Diffusion and DALL-E. While some of these ideas have existed since at least 2015 and theoretical foundations were laid in the 1980s, they became popularized in the 2020 article [SSDK+20]. Here, we summarize some of the ideas of 'score-based' diffusion models and their theoretical underpinning.

Generative AI is often framed as a modern version of the problem of 'sampling' in statistics. For instance, generating an image of a human face in RGB format with resolution 256×256 can be considered as producing a sample from a probability distribution on the corresponding 'space of images,' a space of dimension $256 \cdot 256 \cdot 3 = 196,608$. This probability distribution feels different from the ones we encounter in introductory classes on statistics: it assigns zero probability to most of the space (randomly chosen color values per pixel, constant colors, images of anything but human faces) and in some models, it even gives all probability to a moderate-dimensional 'surface' (say, a few dozen dimensions) in this very high-dimensional ambient space. We are currently interested in the 'support' of the probability distribution as the set of reasonable images of faces, not so much the probabilities it assigns to different sub-classes. Still, this way of modeling allows us to use a variety of well-developed tools below.

Naturally, we have no way of writing a probability distribution on the nearly 200,000-dimensional space of images in any neat way. We cannot hope to find a formula. All we have access to is a dataset, a collection of 'samples' from the true underlying distribution. If we have n data points[1] $x_1, \dots, x_n$, which we interpret as

[1] Note that each 'data point' x_i is a vector in our very high-dimensional space $\mathbb{R}^d$, $d \approx 200,000$!!

samples from the right probability distribution, we can define the 'empirical distribution' of the dataset $\frac{1}{n}\sum_{i=1}^n \delta_{x_i}$ which gives probability $1/n$ to each data point (and zero to the rest of space). This is a coarse proxy, but we can use it in computations.

This perspective of generation as sampling is prevalent in many areas of generative AI. Diffusion models are merely one way in which models can generate samples based on imitating (and reversing) the mechanisms underlying physical diffusion.

In this article, we only assume that the reader is familiar with calculus in dimension $d \geq 3$ (norm, gradient, divergence, Laplacian) and the basics of probability theory (probabilities, expectations, uniform and normal distributions). We therefore focus primarily on general ideas, not technical details or specifics of practical implementation. Of course, to make diffusion models run efficiently on real world data, substantial skill is required beyond theoretic understanding.

Diffusion models combine two ideas: Sampling in statistics and diffusion in physics. We discuss some background information on both in Sections 2 and 3 respectively before describing how to combine them in Section 4.

2. **Sampling in Statistics**

Let us start with a much older version of the sampling problem: *Write a program which emulates a fair coin toss.* If we expect the program to be run once, we can solve this without much coding: We toss a coin and simply tell the program to return the result of that one coin toss. If we expect to run the program often, however, an attentive user will quickly catch on that we are not doing what we claim. So, how do we generate outputs that look like fair coin tosses even if we want many of them?

The problem here is that there is no true randomness in a computer program. Instead, we work with 'pseudo-random' numbers which have the right 'statistical properties' and are unpredictable enough to be useful. For instance, we can fix our favorite irrational number ξ and look at the *non-integer part* of $\xi, 2\xi, 3\xi, \dots$ in the interval from 0 to 1. Statistically, these will be distributed according to the *uniform* distribution (i.e. if we fix any interval between numbers a and b, then the non-integer part of $n\xi$ lies between a and b roughly with frequency $b-a$).[2] In more precise terms,

$$\lim_{n\to\infty} \frac{\#\{k = 1, \dots, n\} : k\xi - \lfloor k\xi \rfloor \in (a,b)\}}{n} = b - a.$$

To emulate fair coin tosses, we could then select an irrational number[3] at programming time, and whenever a user calls the program to generate N coin tosses, we select

[2] If we used $\xi \in \mathbb{Q}$ instead, we would eventually see repetition and entirely miss short intervals.
[3] Or rather, a computer representation of one.

a semi-random starting point k in a fashion that guarantees variety (for instance, number of seconds elapsed since midnight on February 28, 1787) and return the non-integer part of $(k+1)\xi, \dots, (k+N)\xi$. This can be interpreted as heads if it is less than $1/2$ and tails if it is more than $1/2$.

Naturally, technical issues abound. If ξ is much smaller than 1, for instance, this would lead to very long consecutive sequences of heads and tails...

Assume that we have a good way of generating random numbers that look like they are independent and uniformly distributed between 0 and 1. Similar to the way we converted these uniformly distributed random numbers between 0 and 1 into outputs heads and tails, we can generate samples from other probability distributions $\mathbb{P}$ by means of the *cumulative distribution function* Φ of the distribution given by

$$\Phi(t) = \mathbb{P}\big(\text{a random sample } x \text{ from } \mathbb{P} \text{ is below } t\big) = \int_{-\infty}^{t} \rho(x)\, dx$$

where ρ is the *density* of $\mathbb{P}$. For instance, $\rho(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)$ for a standard normal distribution. Since $\Phi' = \rho > 0$ (at least in the example), Φ is strictly monotone increasing and we can find an inverse function of Φ.[4] Then, if x is distributed uniformly between 0 and 1, we have

$$t = \mathbb{P}(x \le t) = \mathbb{P}\big(\Phi^{-1}(x) \le \Phi^{-1}(t)\big) \qquad \Rightarrow \quad \Phi(s) = \mathbb{P}\big(\Phi^{-1}(x) \le s\big)$$

where we used the monotonicity of Φ^{-1} and substituted $t = \Phi(s)$. So, $\Phi^{-1}(x)$ is a random quantity with cumulative distribution function Φ, meaning that if we can compute Φ^{-1}, we can transform uniformly distributed random points between 0 and 1 into random samples from the distribution with the cumulative distribution function Φ.

This allows us to generate samples from many distributions in one dimension and also for instance from the d-dimensional normal distribution (just generate d independent samples from a one-dimensional normal distribution for the d coordinates).

We glossed over much of the detail, but the strategy is clear:

1. Find a mechanism to generate a convincing imitation of independent random samples from one probability distribution $\mathbb{P}$ (uniform on $[0,1]$ in our example).

2. Find a way to transform those samples to something that looks like random samples from a different probability distribution $\mathbb{Q}$ that we are actually interested in (equal probability of 'heads' and 'tails' or normal in the examples above).

[4] The inverse function is defined by the identity $\Phi^{-1}(\Phi(t)) = t$ for all admissible inputs t — think x^2 and $\sqrt{x}$ or $\exp(x) = e^x$ and $\log(x)$.

At a high level, diffusion models are a different implementation of the same strategy in a much more challenging scenario. The 'easy' distribution is a high-dimensional normal distribution which we can sample by using the approach we just discovered. The tricky part is step 2: How do we transform samples from the normal distribution (images with the right shape, but random noise entries for all color channels of all pixels) to a realistic image (a 'sample from the original distribution')?

3. **Stochastic Analysis and Partial Differential Equations**

Differential equations of various kinds (ordinary, partial, stochastic) are among the most popular tools of mathematical modeling across applications. We build a little background that is useful to understand the mechanism of diffusion models by analogy to classical mechanics.

3.1. **Diffusion: Stochastic Particle Dynamics**

In physics, diffusion is the process by which, for instance, ink spreads out in water. If ink is released in a glass of water in one location, then it will over time spread out until it is evenly distributed and we can no longer tell where it originated. This process is *undirected* and stems from particles colliding on the length-scaled of individual molecules. These unobservable collisions lead to ragged, seemingly random paths on larger length scales. A thought experiments suggests that if we divide the glass into two regions[5] and one contains much more ink than the other, then if we let all particles bounce around for some time, every molecule of ink has a certain chance to cross over to the other region. If there are more particles in one region than in the other, we simply expect more crossings from the 'high concentration' region to the 'low concentration' region.

Mathematically, we want to model particle trajectories by a stochastic differential equation for the location $X = (x, y, z)$ of an individual particle like

$$\frac{dX}{dt} = v\big(t, X\big) + N_t$$

where v is the local velocity of the fluid (this plays an important role for instance if we drop ink into the glass from some height). We do not directly model collisions between particles we cannot observe, but rather use some type of 'random noise' N_t. Unfortunately, to make this rigorous with realistic 'Brownian motion' noise $N_t = \sqrt{2}\,\sigma\, dB_t/dt$,

[5] In our head, not physically influencing the glass.

we would need some familiarity with advanced probability theory.[6] However, we can make a good approximation

$$dX_t = v\big(t, X_t\big)\, dt + \sqrt{2}\sigma\, dB_t \qquad \Leftrightarrow \quad X(t+\tau) \approx X(t) + \tau\, v\big(t, X(t)\big) + \sqrt{2\tau}\sigma\, B_t \quad (1)$$

where B_t is a random vector with entries that are all independent samples of a standard normal distribution. The velocity v gives us a 'drift' component and the noise induces oscillations of mean zero with the parameter σ governing the noise intensity.

3.2. **Diffusion: Particle Densities**

Knowing the exact location of every particle of ink is neither possible nor useful to a physicist. Instead, we are interested in 'macroscopic' averages, like the density of the ink in water/the 'hue' of the mixture. The question is, if we know the initial density of ink in the glass and the velocity field v of the mix, how do we find the distribution of ink at a later time? Or, knowing the microscopic physics, how do we predict the behavior of macroscopic or mesoscopic averages?

A simple example lends some insight. For now, let us ignore the fact that the water is in a glass, but allow an ink particle to travel on the entire space. Additionally, we assume that the macroscopic velocity v is zero, meaning that random motion is the only physical process going on. In this case, the probability of finding the particle which started at the origin in a region of space A at time $t > 0$ is given by a normal distribution (on $\mathbb{R}^d$) with mean 0 (i.e. the most likely location to find a particle remains at its starting point) and independent coordinates (movement in x-direction does not affect movement in y-direction). The *variance* is simply given by t. Specifically:

$$\mathbb{P}(x_t \in A) = \int_A \frac{1}{(2\pi t)^{d/2}} \exp\left(-\frac{\|x\|^2}{2t}\right) dx.$$

Experienced mathematicians will recall the density function $\phi(t,x) = \frac{1}{(2\pi t)^{d/2}} e^{-\frac{\|x\|^2}{2t}}$ from a different context: this is the *heat kernel*, a particularly useful solution to the *heat equation*

$$\partial_t u = \Delta u = \big(\partial_{x_1}\partial_{x_1} + \cdots + \partial_{x_n}\partial_{x_n}\big)u.$$

The heat equation is a partial differential equation which models the spread of heat in a medium and — owing to the connections we discussed — the diffusion of one a hydrophilic fluid in aqueous solution. More generally, if particles evolve due to stochastic dynamics $dX_t = v_t\, dt + \sqrt{2}\,\sigma dB_t$ and are initially distributed with some probability

[6] Specifically, Itô's stochastic integration theory. In keeping with the standard notation from this field, we formally multiply the differential equation above by dt.

density p_0, then this density evolves according to the *Fokker-Planck equation*

$$\partial_t p = \sigma^2 \, \Delta p - div(p \, v).$$

We assume that the diffusion coefficient σ is constant throughout the domain for simplicity, but density p and velocity v may of course depend on time t and position x.[7]

3.3. **Particle systems and their densities**

Knowing the location of ink particles tells us the density of ink in water, and knowing the density, we can at least infer how many particles we should expect within a certain area. Similarly, knowing how the particles move tells us how the densities change. The converse is emphatically *not* true: For example, the population of the City of Pittsburgh has declined most years between 2000 and 2025 [Fra25], but this does *not* mean that people only moved away from Pittsburgh during this time period. Similarly, the population of a city holding steady does not mean that people are not moving there or away; it only means that *the rates of change balance*. These simple macroscopic averages may fail to tell us the true dynamics happening.

We call this a 'dynamic equilibrium': Our macroscopic view does not change, but on a smaller scale, people (or particles) are moving around.

4. **Diffusion Models**

Diffusion models combine ideas from sampling and the physical process of diffusion. We start with two observations:

1. We can create convincing 'random' samples from a standard normal distribution, even in high dimension.
2. Under diffusion dynamics $\partial_t \rho = \Delta \rho$, every initial probability distribution ρ_0 eventually approaches a standard normal distribution in the limit $t \to \infty$.

Now, can we reverse the process to go backwards in time from the standard normal distribution to the initial probability distribution? This should enable us to use

[7] The Fokker-Planck equation is also known as the *Kolmogorov forward equation* of the process. The related *Kolmogorov backward equation* governs the time evolution of $u(t,x) = \mathbb{E}[f(X_t) \,|\, X_0 = x]$. We can derive the backward equation from Itô's formula in stochastic analysis and get the forward equation as a consequence by partial integration — this is where the minus sign comes from. For the 'diffusion term,' we integrate by parts twice (no minus), for the 'drift term' we integrate by parts only once. Itô's formula is essentially a chain rule which takes into account stochastic fluctuations as well.

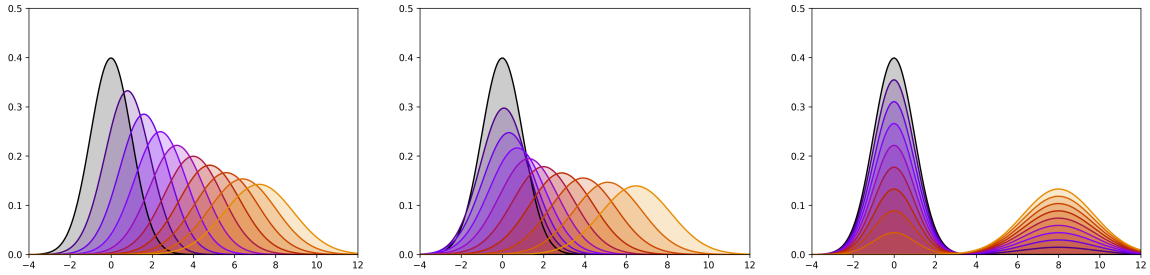

Figure 1: Three different ways to connect normal probability distributions by paths of probability distributions. The first two paths are 'horizontal': every distribution along the path is normal, just the means and variances are evolving. The third path is 'vertical': one distribution disappears, the other one rises. Distributions along the path are Gaussian mixture models. The horizontal paths can be realized by physical 'particle dynamics' as the law of traveling particles, while a 'vertical' path may require particles to teleport or move in an unphysical fashion.
Knowing the path of the probability distributions does not indicate the particle dynamics: They could travel orderly left to right, or they may be bouncing around in a way that 'averages out' suitably.

the same strategy as before and transform sample points drawn from the normal distribution into samples from ρ_0. We visualize paths in the space of probability distributions in Figure 1.

4.1. **A path in the space of probability distributions**

If we deform any probability distribution to the standard normal distribution by the same physical dynamics, reversing this process must be highly unstable. If the endpoint is always the same, it must be hard to work back from there to figure out what the initial input was. We cannot rely on 'general' physics, but we need to integrate some more information about where we came from or along which path we traveled.

Assume that p_0 is any probability density on $\mathbb{R}^d$ and consider the diffusion of p_0 by the partial differential equation

$$\begin{cases} \partial_t p &= \Delta p + \nabla \cdot (p\, x) & \text{for all times } t > 0 \text{ and } x \in \mathbb{R}^d \\ p &= p_0 & \text{at time } t = 0 \text{ and for all } x \in \mathbb{R}^d. \end{cases}$$

Let us consider how the evolution looks 'backwards in time.' When we consider $q(t, x) = p(T - t, x)$ for some fixed time $T > 0$, we have

$$\partial_t q(t, x) = -\partial_t p(T - t, x) = -\Delta p(T - t, x) - \nabla \cdot (p\, x) = -\Delta q(t, x) - \nabla \cdot (q\, x).$$

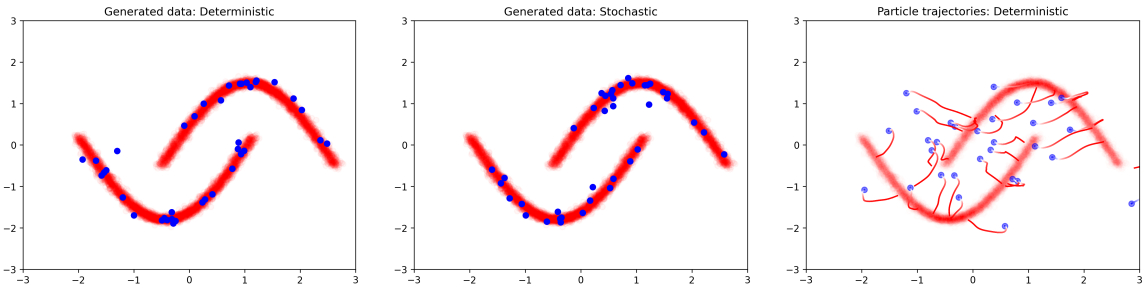

Figure 2: Sampling from a 'two moons' type dataset with $\beta = 0$ (left) and $\beta = 1$ (middle). The terminal time in this simulation was $T = 3$, and the ODE and SDE were resolved with step size $\tau = 0.01$. On the right, we see the trajectories from the initial random point (blue dot) to the region where data lives. The 'backwards' trajectories are deterministic (i.e. $\beta = 0$).

Unlike the heat equation $\partial_t p = \Delta p$, the backwards heat equation $\partial_t q = -\Delta q$ is highly unstable, and solutions tend to 'blow up' in finite time, both in theory and computations. On a physics level, we are trying to 'undo' collisions with small particles that are invisible to us, a task doomed to fail. Luckily, we have an alternative way of writing the equation: note that

$$\Delta q(t,x) = \Delta p(T-t,x) = \nabla\cdot\nabla p = \nabla\cdot\left(p\,\frac{\nabla p}{p}\right) = -\nabla\cdot\big(p\,\nabla\log p\big) = \nabla\cdot\big(q\,\nabla\log p\big)$$

where we evaluate both p and q at the same point x, but different times $T-t$ and t respectively. We can therefore write

$$\partial_t q = \beta\,\Delta q - \nabla\cdot\big(q\,[(1+\beta)\nabla\log p + x]\big) \tag{2}$$

for any constant β (which may even depend on time) — however, the useful regime is $\beta \geq 0$ for which the PDE is stable. The vector-valued function $sc(t,x) = \nabla\log p(t,x)$ of space and time is important enough that it receives its own name: the *score* function. On a philosophical level, keeping track of the score function along the diffusion forwards in time allows us to reverse the dynamics after since it encodes 'where we came from.'

4.2. **Particle trajectories and sampling**

We now have a model to connect the probability distributions: for the forward dynamics, we solve the diffusion equation with drift. For the backward dynamics, we choose some time $T > 0$ at which $p(T,x)$ is close enough to the density of a normal

distribution and a parameter $\beta \geq 0$ and solve the partial differential equation

$$\begin{cases} \partial_t q &= \beta\, \Delta q - \nabla \cdot \big(q\,[(1+\beta)\, sc(T-t,x) + x]\big) & \text{for } 0 < t < T \\ q &= \text{standard normal} & \text{at } t = 0. \end{cases}$$

So far, this is not useful: the probability distributions on the high-dimensional space are inaccessible to us. Luckily, this partial differential equation has a form that is quite familiar to us by now: we can interpret q as the distribution of randomly evolving particles according to probabilistic dynamics evolving by the stochastic differential equation

$$dZ_t = \big(Z_t + (1+\beta)\, sc(T-t, Z_t)\big)\, dt + \sqrt{2\beta}\, dB_t$$

which we can discretize by taking a small time-step of size $\tau > 0$

$$Z_{t+\tau} = Z_t + \tau\big(Z_t + (1+\beta)\, sc(T-t, Z_t)\big) + \sqrt{2\beta\tau}\, B_t$$

with random noise B_t (a standard normal distribution in d dimensions which is independent of everything else). This gives us a whole family of ways to evolve a particle starting at a random location according to a standard normal distribution to a point which, at terminal time T for the backward process, becomes a random sample from the initial distribution p_0.

As we noted earlier, the evolution of the distribution does not determine particle trajectories.[8] We have a diverse family of sampling methods: deterministic ($\beta = 0$) and with varying levels of stochastic dominance (depending on how large we choose $\beta > 0$). Deterministic sampling is easier and produces good individual samples, but stochastic sampling is more likely to truly explore a data distribution. The better method may depend on the application — how fast is a sample needed, and what is the computational budget?

We show samples generated by a diffusion model with $\beta = 0$ (deterministic) and $\beta = 1$ (stochastic) in a simple two-dimensional example in Figures 2 and 3. In all cases, we draw 40 random points and let them evolve by the ordinary differential equation or stochastic differential equation to generate realistic samples.

4.3. **Where does ML come in?**

So far, we have not talked about machine learning. We have an inexact method (since we use a finite terminal time T and small time steps instead of continuous time analysis), but nothing had to be learned.

Diffusion models are considered a method of deep learning because we need to know the score function $sc(t,x) = \nabla \log p(t,x)$, which we need to 'learn,' and the

[8] An excellent visual illustration in a similar context can be found in Figure 1 of [ABVE23].

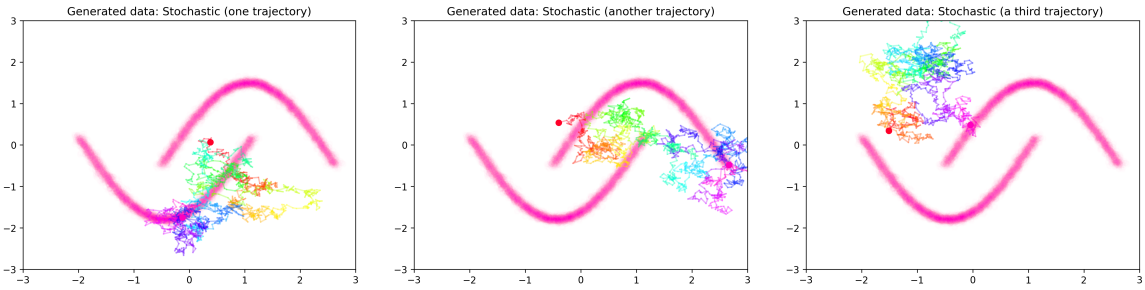

Figure 3: Three stochastic paths for $\beta = 1$. We can tell that the paths explore a larger part of data space. The second trajectory illustrates that even if we start close to the dataset, we may end up with a sample in an entirely different part of the data region.

power of the function class of neural networks makes this possible. In principle, we would have access to an expression for the score function using the dataset as an estimator, but using it directly is infeasible for large datasets.

To 'learn' the score, we would like to select a neural network f with tunable parameters θ and minimize an average discrepancy between the score and the neural network output such as

$$Loss(\theta) = \int_0^T \int_{\mathbb{R}^d} \|f(\theta; t, x) - sc(t, x)\|^2 \, p(t, x) \, dx \, dt.$$

This loss function cannot be evaluated without the score, which is the very object we are trying to learn! Conveniently, this can be reformulated in a way which we can approximate numerically, even without ever solving the forward diffusion! Specifically, we can show that

$$\begin{aligned}
&\int_{\mathbb{R}^d} \|f(\theta; t, x) - sc(t, x)\|^2 \, p(t, x) \, dx \\
&= \frac{1}{(2\pi)^{d/2}} \int_{\mathbb{R}^d} \int_{\mathbb{R}^d} \left\| f\left(\theta;\, t,\, e^{-t} x_0 + \sqrt{1 - e^{-2t}} z\right) + \frac{1}{\sqrt{1 - e^{-2t}}} \, z \right\|^2 p_0(x) \, e^{-\frac{\|z\|^2}{2}} \, dz + C
\end{aligned}$$

for some constant which does not depend on the parameter θ. Because of this, the two minimization problems are equivalent in the sense that they have the same minimizers! This is all we need. In particular, we can proceed as follows: For a fixed 'batch size' k:

- sample times $t_1, \dots, t_k$ (for instance from a uniform distribution on $[0, T]$ for large enough T),
- sample k data points $x_{i(1)}, \dots, x_{i(k)}$ from the original dataset, and

- sample k points $z_1, \ldots, z_k$ from a standard normal distribution on the d-dimensional space $\mathbb{R}^d$.

Use the proxy minimization problem

$$\widetilde{L}(\theta) = \frac{1}{k} \sum_{j=1}^{k} \left\| f\left(\theta; t_j, e^{-t_j} x_{i(j)} + \sqrt{1 - e^{-2t_i}} z_j\right) + \frac{1}{\sqrt{1 - e^{-2t_j}}} z_j \right\|^2$$

to nudge the parameters θ in the right direction. Everything in this problem can be computed efficiently — the trick is to evaluate the neural network at the modified point $e^{-t_j} x_{i(j)} + \sqrt{1 - e^{-2t_i}} z_j$ in the data domain. Repeating this over many iterations with many different samples, we perform a 'stochastic gradient descent' to train the parameters of our neural network so that its outputs are a good approximation of the true score function. This is the only place where deep learning comes in — diffusion models are not inherently neural-network based, but in practice, we need their approximation power.

For the reader's convenience, we explain the equivalence of the minimization problems in more detail in the Appendix, based on [CCL+22, Appendix A].

4.4. **An open question**

Diffusion models are a fairly recent addition to the toolbox of machine learning, and we still do not fully understand details of their inner workings. For instance, if we were to *exactly* solve the backwards dynamics for sampling, then we would end up with a sample from the original distribution, which in our case is the 'empirical distribution' of the dataset. In other words, we would just randomly select one of our data points as the random sample.

Evidently, this is not what is happening in diffusion models which can generate authentic new data. However, why we make a large enough error in the backwards dynamics to produce a novel sample, but a small enough error to produce a reasonable sample, remains mysterious.

5. **Summary**

Diffusion models are a tool of generative AI which takes inspiration from the physical process of diffusion and finds a way of reversing it *on the distribution level*. For many of us, the first introduction to diffusion models we find uses images like Figure 4. *This kind of image is a lie:* reversing diffusion on the particle level is impossible since the forward diffusion is stochastic. If we find a piece of pigment in

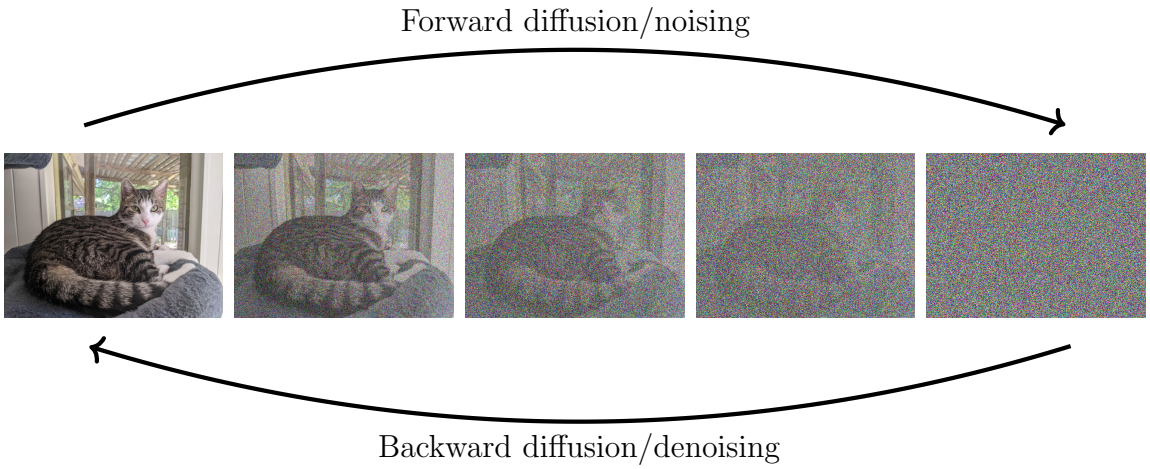

Figure 4: This image is an inaccurate simplification: we cannot reverse particle trajectories/the noising of images – what we can do is design processes which take noise as input and generate 'reasonable' images as output.

the glass of water after some time, there is no way of reconstructing where it started its journey.

On the other hand, it is possible to find (unphysical) dynamics with a velocity field stemming from the score function along the forward diffusion that allows us to reverse the path of the ink cloud. These distribution level dynamics can be interpreted as (deterministic or stochastic) trajectories of particles — however, they are *not* the paths of diffusing particles backwards in time. A more accurate depiction is given in Figure 5.

This perspective of thinking about distributional dynamics which can be discretized by particle dynamics is also the basis for a general class of models which takes a step away from physical diffusion dynamics, but gives the user additional flexibility [ABVE23].

6. **How do I learn more?**

One of the best places to become familiar with stochastic differential equations is in classes on Financial Mathematics, where asset prices are modeled by stochastic differential equations. While the details vary between applications, the foundations are the same for applications in statistical mechanics, financial markets, and generative AI models.

At a high level, diffusion models combine ideas from stochastic analysis, partial

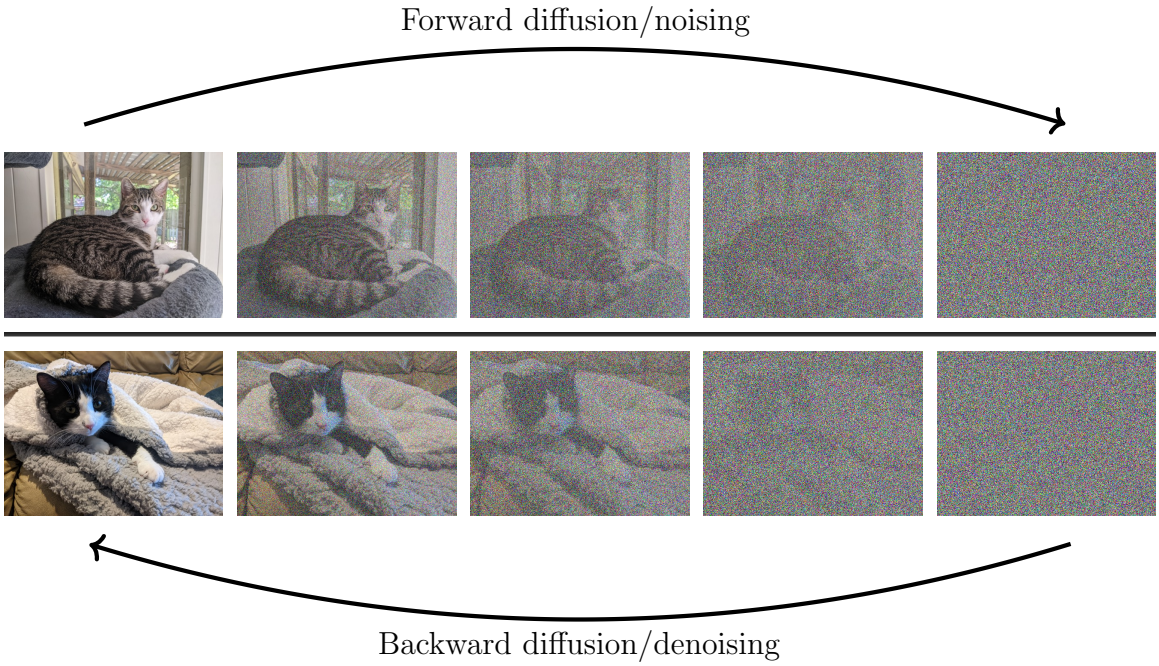

Figure 5: This figure is more accurate. We can reverse the process by which all images are mapped to noise and find multiple processes which take random noise as a starting point and generate a reasonable picture. However, at least the forward diffusion is stochastic (and also the backwards process in some models). There is no way of undoing the noising process *per image*, and we do not have fine control over *which* image is generated.
Curiously, the *backward* process may be deterministic and in a way more regular while the forward process is *always* stochastic.

differential equations, and statistical mechanics with machine learning models. By necessity, we glossed over many of the details. For any program in applied mathematics, learning about differential equations, probability theory, and statistics to fill in the missing parts is a great idea. Learning numerical methods and practicing coding skills is essentially to put theory into practice.

Conceptually, diffusion models are a tour de force. We take inspiration from classical physics for an extremely high-dimensional application in data science and seamlessly pass back and forth between a PDE model for the density and an SDE model on the particle level. The foundational underpinning is a way to connect to probability distributions by a path in the (infinite-dimensional and non-linear) space of probability distributions. Another useful topic to learn more about the admittedly abstract underpinnings of this and other classes of generative AI models (Wasserstein-GANs, flow matching) is therefore the theory of optimal transport.

Diffusion models are a fairly new method of machine learning and still the subject of active research. While they are covered in newer textbooks, any source published before 2022 is unlikely to contain much (or any) information on them. In a rapidly evolving field, perspectives shift, and at the fore-front of research, ideas always outpace introductions. This makes it crucial to focus on understanding concepts and foundations: benchmarks may fall, and models will change, but concepts last – and every once in a while, knowledge in an unrelated field of physics may inspire a powerful new method in a totally different field like AI.

A. **A computable loss function minimized by the score**

As a starting point, assume that we want to minimize a time average of

$$Loss_t(\theta) = \int_{\mathbb{R}^d} \|f(\theta; t, x) - sc(t, x)\|^2 \, p(t, x) \, dx \tag{3}$$

where $p(t, x)$ is the probability density of the evolving particles – in other words, we count the discrepancy between the output $f(\theta; t, x)$ of our neural network with parameters θ and the true score function $sc(t, x)$ more in regions of higher probability, and we are okay with larger errors in the output in regions we don't expect to visit anyways. Other choices would be possible, but this one will turn out to have good properties. A time average could be

$$Loss(\theta) = \int_0^T Loss_t(\theta) \, dt \qquad \text{or} \quad Loss(\theta) = \int_0^\infty Loss_t(\theta) \, e^{-t} \, dt$$

or similar – the differences are not important for us and we focus on fixed time t. Our goal is to find a function $\widetilde{Loss}_t$ which is computable – this is important since our

entire problem is that we do not know sc – and has the same minimizers as (3).

We start by expanding the square:

$$Loss_t(\theta) = \int_{\mathbb{R}^d} \big(\|f(\theta;t,x)\|^2 - 2f(\theta;t,x)\cdot sc(t,x)\big)\, p(t,x)\, dx + \underbrace{\int_{\mathbb{R}^d} \|sc(t,x)\|^2\, p(t,x)\, dx}_{=:C}$$

The constant C does not depend on θ, for the purposes of minimization problems we can ignore it and focus on the left two terms. We will leave the first one alone as it already has the right form and focus on the cross term instead. Recall that

$$sc(t,x) = \nabla \log(p(t,x)) = \frac{1}{p(t,x)}\,\nabla p(t,x) \quad \Rightarrow \quad f(\theta;t,x)\cdot sc(t,x)\, p(t,x) = f\cdot \nabla p.$$

We now use the product rule for the divergence operator and the divergence theorem from Calculus 3 for integration by parts. We do this on the whole space and without boundary terms (we can show that under reasonable assumptions $p(t,x)$ goes to zero as $|x|\to\infty$ so fast that the boundary terms disappear if we apply the theorem on larger and larger balls).

$$\begin{aligned}\int_{\mathbb{R}^d} f(\theta;t,x)\cdot sc(t,x)\, p(t,x)\, dx &= \int_{\mathbb{R}^d} f(\theta;t,x)\cdot \nabla p(t,x)\, dx \\ &= \int_{\mathbb{R}^d} div\big(p(t,x)\, f(\theta;t,x)\big) - p(t,x)\, div\big(f(\theta;t,x)\big)\, dx \\ &= -\int_{\mathbb{R}^d} div_x\big(f(\theta;t,x)\big)\, p(t,x)\, dx.\end{aligned}$$

So, an equivalent minimization problem that does not involve the score function directly would be for

$$\widetilde{\widetilde{Loss}}_t(\theta) = \int_{\mathbb{R}^d} \big(\|f(\theta;t,x)\|^2 + 2\, div\big(f(\theta;t,x)\big)\big)\, p(t,x)\, dx.$$

However, this problem is still costly to solve since it involves the derivatives of f with respect to all d coordinate entries of f, which is very time-consuming to compute for neural networks. Even worse, we need to integrate with respect to the probability density $p(t,x)$, which requires us to solve a partial differential equation... We therefore keep looking for a better equivalent loss.

Let us turn our attention to $p(t,x)$ – this will allow us to solve both problems at once. Using PDE theory or stochastic analysis, we find that

$$\int_{\mathbb{R}^d} div_x\big(f(\theta;t,x)\big)\, p(t,x)\, dx = \int_{\mathbb{R}^d} div_x\big(f(\theta;t,\, e^{-t}x' + \sqrt{1-e^{-2t}}z)\big)\, p_0(x')\, dx'\, \frac{e^{-\frac{\|z\|^2}{2}}}{(2\pi)^{d/2}}\, dz$$

where the point x is replaced by a weighted sum between a normally distributed point

z and a point x' drawn from the initial distribution p_0. Fully deriving the expression goes a bit beyond this article, but we note the similarity to the heat equation without the drift term, where solutions can be written as averages of u_0 weighted by suitable normal distributions.

For the integral of a divergence with respect to a normal distribution, we recall that

$$\begin{aligned}\int_{\mathbb{R}^d} div(f(z))\,\frac{1}{(2\pi)^{d/2}}\,e^{-\|z\|^2/2}\,dz &= -\int_{\mathbb{R}^d} f(z)\cdot\nabla e^{-\|z\|^2/2}\frac{1}{(2\pi)^{d/2}}\,dz\\ &= \int_{\mathbb{R}^d} f(z)\cdot z\,\frac{1}{(2\pi)^{d/2}}e^{-\|z\|^2/2}\,dz\end{aligned}$$

or in simpler terms

$$\int_{\mathbb{R}^d} div(f(z))\,d\gamma = \int_{\mathbb{R}^d} f(z)\cdot z\,d\gamma$$

where we use the shorthand $d\gamma$ for the integral with respect to the 'Gaussian' (normal) distribution as above. Thus we can finally rewrite

$$\begin{aligned}\int_{\mathbb{R}^d} div_x\big(f(\theta;t,x)\big)\,p(t,x)\,dx &= \int_{\mathbb{R}^d}\int_{\mathbb{R}^d} div\big(f\big(\theta;t,e^{-t}x+\sqrt{1-e^{-2t}}z\big)\big)\,p_0(x)\,d\gamma_z\,dx\\ &= \int_{\mathbb{R}^d}\int_{\mathbb{R}^d} f\big(\theta;t,e^{-t}x+\sqrt{1-e^{-2t}}z\big)\cdot\frac{1}{\sqrt{1-e^{-2t}}}\,z\,d\gamma_z\,p_0(x)\,dx.\end{aligned}$$

The factor in front of the divergence comes from the chain rule $\big(div(f)\big)(\alpha x) = \frac{1}{\alpha}\,div(f(\alpha x))$. We have thus eliminated the divergence from the computation at the cost of a double integral in place of a single integral. Given that we only approximate integrals by computing averages over batches of data points and random Gaussian noise for z (a variant of Monte-Carlo integration), this is a minor cost in practice. Overall, we find that

$$\begin{aligned}Loss_t(\theta) &= \int_{\mathbb{R}^d}\big(\|f(\theta;t,x)\|^2 - 2f(\theta;t,x)\cdot sc(t,x)\big)\,p(t,x)\,dx + C\\ &= \int_{\mathbb{R}^d}\int_{\mathbb{R}^d}\left(\|f(\theta;t,\xi(t,x,z))\|^2 - 2f(\theta;t,\xi(t,x,z))\cdot\frac{z}{\sqrt{1-e^{-2t}}}\right)d\gamma_z\,p_0(x)\,dx + C\\ &= \int_{\mathbb{R}^d}\int_{\mathbb{R}^d}\left\|f(\theta;t,\xi(t,x,z)) - \frac{z}{\sqrt{1-e^{-2t}}}\right\|^2 d\gamma_z\,p_0(x)\,dx + \widetilde{C}\end{aligned}$$

for another constant $\widetilde{C}$ from completing another square – again, the constant does not depend on θ and is therefore irrelevant to our minimization problem. The equivalent loss function we use is

$$\widetilde{Loss_t}(\theta) = \int_{\mathbb{R}^d}\int_{\mathbb{R}^d}\left\|f(\theta;t,\xi(t,x,z)) - \frac{z}{\sqrt{1-e^{-2t}}}\right\|^2 d\gamma_z\,p_0(x)\,dx$$

which we can estimate efficiently by producing data samples as outlined above. This equivalence enables us to use diffusion models in practice.

References

[ABVE23] Michael S Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

[CCL+22] Sitan Chen, Sinho Chewi, Jerry Li, Yuanzhi Li, Adil Salim, and Anru R Zhang. Sampling is as easy as learning the score: Theory for diffusion models with minimal data assumptions. *arXiv preprint arXiv:2209.11215*, 2022.

[Fra25] Julia Fraser. City of Pittsburgh population grew in 2024, adding nearly 3,000 new residents. *90.5 WESA*, 15 May 2025.

[SSDK+20] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Stephan Wojtowytsch
Department of Mathematics, University of Pittsburgh, Pittsburgh, PA
Email: s.woj@pitt.edu