Step-by-Step Diffusion: An Elementary Tutorial

Preetum Nakkiran[1], *Arwen Bradley*[1], *Hattie Zhou*[1,2], *Madhu Advani*[1]

[1]*Apple*, [2]*Mila, Université de Montréal*

We present an accessible first course on diffusion models and flow matching for machine learning, aimed at a technical audience with no diffusion experience. We try to simplify the mathematical details as much as possible (sometimes heuristically), while retaining enough precision to derive correct algorithms.

Contents

1 Fundamentals of Diffusion **3**
- 1.1 Gaussian Diffusion . . . 3
- 1.2 Diffusions in the Abstract . . . 5
- 1.3 Discretization . . . 6

2 Stochastic Sampling: DDPM **8**
- 2.1 Correctness of DDPM . . . 9
- 2.2 Algorithms . . . 11
- 2.3 Variance Reduction: Predicting x_0 . . . 11
- 2.4 Diffusions as SDEs [Optional] . . . 13

3 Deterministic Sampling: DDIM **16**
- 3.1 Case 1: Single Point . . . 16
- 3.2 Velocity Fields and Gases . . . 18
- 3.3 Case 2: Two Points . . . 18
- 3.4 Case 3: Arbitrary Distributions . . . 20
- 3.5 The Probability Flow ODE [Optional] . . . 21
- 3.6 Discussion: DDPM vs DDIM . . . 22
- 3.7 Remarks on Generalization . . . 23

4 Flow Matching **25**
- 4.1 Flows . . . 25
- 4.2 Pointwise Flows . . . 26
- 4.3 Marginal Flows . . . 26
- 4.4 A Simple Choice of Pointwise Flow . . . 27
- 4.5 Flow Matching . . . 28
- 4.6 DDIM as Flow Matching [Optional] . . . 30
- 4.7 Additional Remarks and References [Optional] . . . 31

5 Diffusion in Practice **32**

A Additional Resources **36**

B Omitted Derivations **38**

Preface

There are many existing resources for learning diffusion models. Why did we write another? Our goal was to teach diffusion as simply as possible, with minimal mathematical and machine learning prerequisites, but in enough detail to reason about its correctness. Unlike most tutorials on this subject, we take neither a Variational Auto Encoder (VAE) nor an Stochastic Differential Equations (SDE) approach. In fact, for the core ideas we will not need any SDEs, Evidence-Based-Lower-Bounds (ELBOs), Langevin dynamics, or even the notion of a score. The reader need only be familiar with basic probability, calculus, linear algebra, and multivariate Gaussians. The intended audience for this tutorial is technical readers at the level of at least advanced undergraduate or graduate students, who are learning diffusion for the first time and want a mathematical understanding of the subject.

This tutorial has five parts, each relatively self-contained, but covering closely related topics. Section 1 presents the fundamentals of diffusion: the problem we are trying to solve and an overview of the basic approach. Sections 2 and 3 show how to construct a stochastic and deterministic diffusion sampler, respectively, and give intuitive derivations for why these samplers correctly reverse the forward diffusion process. Section 4 covers the closely-related topic of Flow Matching, which can be thought of as a generalization of diffusion that offers additional flexibility (including what are called rectified flows or linear flows). Finally, in Section 5 we return to diffusion and connect this tutorial to the broader literature while highlighting some of the design choices that matter most in practice, including samplers, noise schedules, and parametrizations.

Acknowledgements

We are grateful for helpful feedback and suggestions from many people, in particular: Josh Susskind, Eugene Ndiaye, Dan Busbridge, Sam Power, De Wang, Russ Webb, Sitan Chen, Vimal Thilak, Etai Littwin, Chenyang Yuan, Alex Schwing, Miguel Angel Bautista Martin, and Dilip Krishnan.

1 Fundamentals of Diffusion

THE GOAL of generative modeling is: given i.i.d. samples from some unknown distribution $p^*(x)$, construct a sampler for (approximately) the same distribution. For example, given a training set of dog images from some underlying distribution p_{dog}, we want a method of producing new images of dogs from this distribution.

One way to solve this problem, at a high level, is to learn a transformation from some easy-to-sample distribution (such as Gaussian noise) to our target distribution p^*. Diffusion models offer a general framework for learning such transformations. The clever trick of diffusion is to reduce the problem of sampling from distribution $p^*(x)$ into to a sequence of *easier* sampling problems.

This idea is best explained via the following Gaussian diffusion example. We'll sketch the main ideas now, and in later sections we will use this setup to derive what are commonly known as the *DDPM* and *DDIM* samplers[1], and reason about their correctness.

[1] These stand for Denoising Diffusion Probabilistic Models (DDPM) and Denoising Diffusion Implicit Models (DDIM), following Ho et al. [2020] and Song et al. [2021].

1.1 Gaussian Diffusion

For Gaussian diffusion, let x_0 be a random variable in $\mathbb{R}^d$ distributed according to the target distribution p^* (e.g., images of dogs). Then construct a sequence of random variables $x_1, x_2, \ldots, x_T$, by successively adding independent Gaussian noise with some small scale σ:

$$x_{t+1} := x_t + \eta_t, \quad \eta_t \sim \mathcal{N}(0, \sigma^2). \tag{1}$$

This is called the *forward process*[2], which transforms the data distribution into a noise distribution. Equation (1) defines a joint distribution over all $(x_0, x_1, \ldots, x_T)$, and we let $\{p_t\}_{t\in[T]}$ denote the marginal distributions of each x_t. Notice that at large step count T, the distribution p_T is nearly Gaussian[3], so we can approximately sample from p_T by just sampling a Gaussian.

[2] One benefit of using this particular forward process is computational: we can directly sample x_t given x_0 in constant time.

[3] Formally, p_T is close in KL divergence to $\mathcal{N}(0, T\sigma^2)$, assuming p_0 has bounded moments.

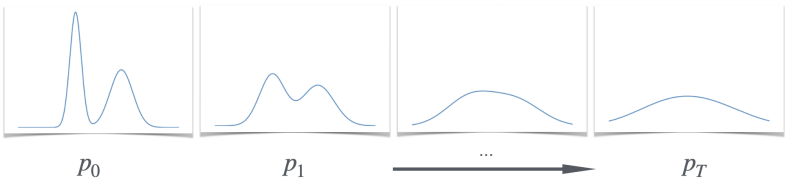

Figure 1: Probability distributions defined by diffusion forward process on one-dimensional target distribution p_0.

Now, suppose we can solve the following subproblem:

> *"Given a sample marginally distributed as p_t, produce a sample marginally distributed as p_{t-1}"*.

We will call a method that does this a *reverse sampler*[4], since it tells us how to sample from p_{t-1} assuming we can already sample from p_t. If we had a reverse sampler, we could sample from our target p_0 by simply starting with a Gaussian sample from p_T, and iteratively applying the reverse sampling procedure to get samples from $p_{T-1}, p_{T-2}, \ldots$ and finally $p_0 = p^*$.

The key insight of diffusion is, **learning to reverse each intermediate step can be easier** than learning to sample from the target distribution in one step[5]. There are many ways to construct reverse samplers, but for concreteness let us first see the standard diffusion sampler which we will call the *DDPM sampler*[6].

The *Ideal DDPM sampler* uses the obvious strategy: At time t, given input z (which is promised to be a sample from p_t), we output a sample from the conditional distribution

$$p(x_{t-1} \mid x_t = z). \tag{2}$$

This is clearly a correct reverse sampler. The problem is, it requires learning a generative model for the conditional distribution $p(x_{t-1} \mid x_t)$ for every x_t, which could be complicated. But if the per-step noise σ is sufficiently small, then it turns out this conditional distribution becomes simple:

Fact 1 (Diffusion Reverse Process). *For small σ, and the Gaussian diffusion process defined in (1), the conditional distribution $p(x_{t-1} \mid x_t)$ is itself close to Gaussian. That is, for all times t and conditionings $z \in \mathbb{R}^d$, there exists some mean parameter $\mu \in \mathbb{R}^d$ such that*

$$p(x_{t-1} \mid x_t = z) \approx \mathcal{N}(x_{t-1};\ \mu,\ \sigma^2). \tag{3}$$

This is not an obvious fact; we will derive it in Section 2.1. This fact enables a drastic simplification: instead of having to learn an

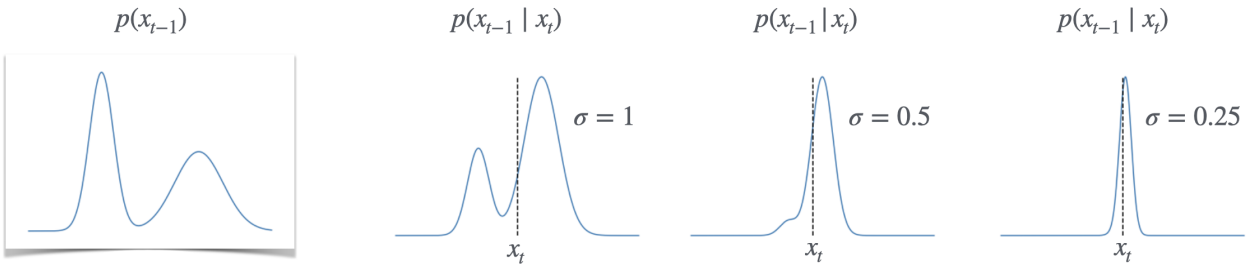

Figure 2: Illustration of Fact 1. The prior distribution $p(x_{t-1})$, leftmost, defines a joint distribution (x_{t-1}, x_t) where $p(x_t \mid x_{t-1}) = \mathcal{N}(0, \sigma^2)$. We plot the reverse conditional distributions $p(x_{t-1} \mid x_t)$ for a fixed conditioning x_t, and varying noise levels σ. Notice these distributions become close to Gaussian for small σ.

[4] Reverse samplers will be formally defined in Section 1.2 below.

[5] Intuitively this is because the distributions (p_{t-1}, p_t) are already quite close, so the reverse sampler does not need to do much.

[6] This is the sampling strategy originally proposed in Sohl-Dickstein et al. [2015].

arbitrary distribution $p(x_{t-1} \mid x_t)$ from scratch, we now know everything about this distribution except its mean, which we denote[7] $\mu_{t-1}(x_t)$. The fact that we can approximate the posterior distribution as Gaussian when σ is sufficiently small is illustrated in Fig 2. This is an important point, so to re-iterate: for a given time t and conditioning value x_t, learning the mean of $p(x_{t-1} \mid x_t)$ is sufficient to learn the full conditional distribution $p(x_{t-1} \mid x_t)$.

[7] We denote the mean as a function $\mu_{t-1} : \mathbb{R}^d \to \mathbb{R}^d$ because the mean of $p(x_{t-1} \mid x_t)$ depends on the time t as well as the conditioning x_t, as described in Fact 1.

Learning the mean of $p(x_{t-1} \mid x_t)$ is a much simpler problem than learning the full conditional distribution, because we can solve it by regression. To elaborate, we have a joint distribution (x_{t-1}, x_t) from which we can easily sample, and we would like to estimate $\mathbb{E}[x_{t-1} \mid x_t]$. This can be done by optimizing a standard regression loss[8]:

$$\mu_{t-1}(z) := \mathbb{E}[x_{t-1} \mid x_t = z] \tag{4}$$

$$\implies \mu_{t-1} = \operatorname*{argmin}_{f:\mathbb{R}^d \to \mathbb{R}^d} \mathop{\mathbb{E}}_{x_t, x_{t-1}} ||f(x_t) - x_{t-1}||_2^2 \tag{5}$$

$$= \operatorname*{argmin}_{f:\mathbb{R}^d \to \mathbb{R}^d} \mathop{\mathbb{E}}_{x_{t-1}, \eta} ||f(x_{t-1} + \eta_t) - x_{t-1})||_2^2, \tag{6}$$

[8] Recall the generic fact that for any distribution over (x, y), we have: $\operatorname{argmin}_f \mathbb{E}\,||f(x) - y||^2 = \mathbb{E}[y \mid x]$

where the expectation is taken over samples x_0 from our target distribution p^*.[9] This particular regression problem is well-studied in certain settings. For example, when the target p^* is a distribution on images, then the corresponding regression problem (Equation 6) is exactly an *image denoising objective*, which can be approached with familiar methods (e.g. convolutional neural networks).

[9] Notice that we simulate samples of (x_{t-1}, x_t) by adding noise to the samples of x_0, as defined in Equation 1.

Stepping back, we have seen something remarkable: we have reduced the problem of learning to sample from an arbitrary distribution to the standard problem of regression.

1.2 *Diffusions in the Abstract*

Let us now abstract away the Gaussian setting, to define diffusion-like models in a way that will capture their many instantiations (including deterministic samplers, discrete domains, and flow-matching).

Abstractly, here is how to construct a diffusion-like generative model: We start with our target distribution p^*, and we pick some base distribution $q(x)$ which is easy to sample from, e.g. a standard Gaussian or i.i.d bits. We then try to construct a sequence of distributions which interpolate between our target p^* and the base distribution q. That is, we construct distributions

$$p_0\ ,\ p_1\ ,\ p_2\ ,\ \ldots, p_T, \tag{7}$$

such that $p_0 = p^*$ is our target, $p_T = q$ the base distribution, and adjacent distributions (p_{t-1}, p_t) are marginally "close" in some appropriate sense. Then, we learn a *reverse sampler* which transforms distributions p_t to p_{t-1}. This is the key learning step, which presumably is made easier by the fact that adjacent distributions are "close." Formally, reverse samplers are defined below.

Definition 1 (Reverse Sampler)**.** *Given a sequence of marginal distributions p_t, a* reverse sampler *for step t is a potentially stochastic function F_t such that if $x_t \sim p_t$, then the marginal distribution of $F_t(x_t)$ is exactly p_{t-1}:*

$$\{F_t(z) : z \sim p_t\} \equiv p_{t-1}. \tag{8}$$

There are many possible reverse samplers[10], and it is even possible to construct reverse samplers which are *deterministic*. In the remainder of this tutorial we will see three popular reverse samplers more formally: the *DDPM* sampler discussed above (Section 2.1), the *DDIM* sampler (Section 3), which is deterministic, and the family of *flow-matching models* (Section 4), which can be thought of as a generalization of DDIM.[11]

[10] Notice that none of this abstraction is specific to the case of Gaussian noise—in fact, it does not even require the concept of "adding noise". It is even possible to instantiate in discrete settings, where we consider distributions p^* over a finite set, and define corresponding "interpolating distributions" and reverse samplers.

[11] Given a set of marginal distributions $\{p_t\}$, there are many possible joint distributions consistent with these marginals (such joint distributions are called *couplings*). There is therefore no canonical reverse sampler for a given set of marginals $\{p_t\}$ — we are free to chose whichever coupling is most convenient.

1.3 Discretization

Before we proceed further, we need to be more precise about what we mean by adjacent distributions p_t, p_{t-1} being "close". We want to think of the sequence $p_0, p_1, \ldots, p_T$ as the discretization of some (well-behaved) time-evolving function $p(x, t)$, that starts from the target distribution p_0 at time $t = 0$ and ends at the noisy distribution p_T at time $t = 1$:

$$p(x, k\Delta t) = p_k(x), \quad \text{where } \Delta t = \frac{1}{T}. \tag{9}$$

The number of steps T controls the fineness of the discretization (hence the closeness of adjacent distributions).[12]

[12] This naturally suggests taking the continuous-time limit, which we discuss in Section 2.4, though it is not needed for most of our arguments.

In order to ensure that the variance of the final distribution, p_T, is independent of the number of discretization steps, we also need to be more specific about the variance of each increment. Note that if $x_k = x_{k-1} + \mathcal{N}(0, \sigma^2)$, then $x_T \sim \mathcal{N}(x_0, T\sigma^2)$. Therefore, we need to scale the variance of each increment by $\Delta t = 1/T$, that is, choose

$$\sigma = \sigma_q \sqrt{\Delta t}, \tag{10}$$

where σ_q^2 is the desired terminal variance. This choice ensures that the variance of p_T is always σ_q^2, regardless of T. (The $\sqrt{\Delta t}$ scaling will turn out to be important in our arguments for the correctness of our reverse solvers in the next chapter, and also connects to the SDE formulation in Section 2.4.)

At this point, it is convenient to adjust our notation. From here on, t will represent a continuous-value in the interval $[0, 1]$ (specifically, taking one of the values $0, \Delta t, 2\Delta t, \ldots, T\Delta t = 1$). Subscripts will indicate *time* rather than *index*, so for example x_t will now denote x at a discretized time t. That is, Equation 1 becomes:

$$x_{t+\Delta t} := x_t + \eta_t, \quad \eta_t \sim \mathcal{N}(0, \sigma_q^2 \Delta t), \tag{11}$$

which also implies that

$$x_t \sim \mathcal{N}(x_0, \sigma_t^2), \quad \text{where } \sigma_t := \sigma_q \sqrt{t}, \tag{12}$$

since the total noise added up to time t (i.e. $\sum_{\tau \in \{0, \Delta t, 2\Delta t, \ldots, t-\Delta t\}} \eta_\tau$) is also Gaussian with mean zero and variance $\sum_\tau \sigma_q^2 \Delta t = \sigma_q^2 t$.

2 Stochastic Sampling: DDPM

In this section we review the DDPM-like reverse sampler discussed in Section 1, and heuristically prove its correctness. This sampler is conceptually the same as the sampler popularized in *Denoising Diffusion Probabilistic Models* (DDPM) by Ho et al. [2020] and originally introduced by Sohl-Dickstein et al. [2015], when adapted to our simplified setting. However, a word of warning for the reader familiar with Ho et al. [2020]: Although the overall strategy of our sampler is identical to Ho et al. [2020], certain technical details (like constants, etc) are slightly different[13].

We consider the setup from Section 1.3, with some target distribution p^* and the joint distribution of noisy samples $(x_0, x_{\Delta t}, \ldots, x_1)$ defined by Equation (11). The DDPM sampler will require estimates of the following conditional expectations:

$$\mu_t(z) := \mathbb{E}[x_t \mid x_{t+\Delta t} = z]. \tag{13}$$

This is a set of functions $\{\mu_t\}$, one for every time step $t \in \{0, \Delta t, \ldots, 1 - \Delta t\}$. In the *training* phase, we estimate these functions from i.i.d. samples of x_0, by optimizing the denoising regression objective

$$\mu_t = \operatorname*{argmin}_{f:\mathbb{R}^d \to \mathbb{R}^d} \mathop{\mathbb{E}}_{x_t, x_{t+\Delta t}} ||f(x_{t+\Delta t}) - x_t||_2^2 \,, \tag{14}$$

typically with a neural-network[14] parameterizing f. Then, in the *inference* phase, we use the estimated functions in the following reverse sampler.

Algorithm 1: Stochastic Reverse Sampler (DDPM-like)

For input sample x_t, and timestep t, output:

$$\widehat{x}_{t-\Delta t} \leftarrow \mu_{t-\Delta t}(x_t) + \mathcal{N}(0, \sigma_q^2 \Delta t) \tag{15}$$

To actually generate a sample, we first sample x_1 as an isotropic Gaussian $x_1 \sim \mathcal{N}(0, \sigma_q^2)$, and then run the iteration of Algorithm 1 down to $t = 0$, to produce a generated sample $\widehat{x}_0$. (Recall that in our discretized notation (12), x_1 is the fully-noised terminal distribution, and the iteration takes steps of size Δt.) Explicit pseudocode for these algorithms are given in Section 2.2.

We want to reason about correctness of this entire procedure: why does iterating Algorithm 1 produce a sample from [approximately] our target distribution p^*? The key missing piece is, we need to prove some version of Fact 1: that the true conditional $p(x_{t-\Delta t} \mid x_t)$ can be well-approximated by a Gaussian, and this approximation gets better as we scale $\Delta t \to 0$.

[13] For the experts, the main difference is we use the "Variance Exploding" diffusion forward process. We also use a constant noise schedule, and we do not discuss how to parameterize the predictor ("predicting x_0 vs. x_{t-1} vs. noise η"). We elaborate on the latter point in Section 2.3.

[14] In practice, it is common to share parameters when learning the different regression functions $\{\mu_t\}_t$, instead of learning a separate function for each timestep independently. This is usually implemented by training a model f_θ that accepts the time t as an additional argument, such that $f_\theta(x_t, t) \approx \mu_t(x_t)$.

2.1 Correctness of DDPM

Here is a more precise version of Fact 1, along with a heuristic derivation. This will complete the argument that Algorithm 1 is correct— i.e. that it approximates a valid reverse sampler in the sense of Definition 1.

Claim 1 (Informal)**.** *Let $p_{t-\Delta t}(x)$ be an arbitrary, sufficiently-smooth density over $\mathbb{R}^d$. Consider the joint distribution of $(x_{t-\Delta t}, x_t)$, where $x_{t-\Delta t} \sim p_{t-\Delta t}$ and $x_t \sim x_{t-\Delta t} + \mathcal{N}(0, \sigma_q^2 \Delta t)$. Then, for sufficiently small Δt, the following holds. For all conditionings $z \in \mathbb{R}^d$, there exists μ_z such that:*

$$p(x_{t-\Delta t} \mid x_t = z) \approx \mathcal{N}(x_{t-\Delta t};\ \mu_z\ ,\ \sigma_q^2 \Delta t). \tag{16}$$

for some constant μ_z depending only on z. Moreover, it suffices to take[15]

$$\mu_z := \mathop{\mathbb{E}}_{(x_{t-\Delta t}, x_t)} [x_{t-\Delta t} \mid x_t = z] \tag{17}$$

$$= z + (\sigma_q^2 \Delta t) \nabla \log p_t(z), \tag{18}$$

where p_t is the marginal distribution of x_t.

[15] Experts will recognize this mean as related to the *score*. In fact, Tweedie's formula implies that this mean is exactly correct even for large Δt, with no approximation required. That is, $\mathbb{E}[x_{t-\Delta t} \mid x_t = z] = z + \sigma_q^2 \Delta t \nabla \log p_t(z)$. The distribution $p(x_{t-\Delta t} \mid x_t)$ may deviate from Gaussian, however, for larger σ.

Before we see the derivation, a few remarks: Claim 1 implies that to sample from $x_{t-\Delta t}$, it suffices to first sample from x_t, then sample from a *Gaussian* distribution centered around $\mathbb{E}[x_{t-\Delta t} \mid x_t]$. This is exactly what DDPM does, in Equation (15). Finally, in these notes we will not actually need the expression for μ_z in Equation (18); it is enough for us know that such a μ_z exists, so we can learn it from samples.

Proof of Claim 1 (Informal). Here is a heuristic argument for why the score appears in the reverse process. We will essentially just apply Bayes rule and then Taylor expand appropriately. We start with Bayes rule:

$$p(x_{t-\Delta t} | x_t) = p(x_t | x_{t-\Delta t}) p_{t-\Delta t}(x_{t-\Delta t}) / p_t(x_t) \tag{19}$$

Then take logs of both sizes. Throughout, we will drop any additive constants in the log (which translate to normalizing factors), and drop all terms of order $\mathcal{O}(\Delta t)$ [16]. Note that we should think of x_t as a constant in this derivation, since we want to understand the

[16] Note that $x_{t+1} - x_t \sim \mathcal{O}(\sqrt{\Delta t})$. Dropping $\mathcal{O}(\Delta t)$ terms means dropping $(x_{t+1} - x_t)^2 \sim \mathcal{O}(\Delta t)$ in the expansion of $p_t(x_t)$, but keeping $\frac{1}{2\sigma_q^2 \Delta t}(x_{t+1} - x_t)^2 \sim \mathcal{O}(1)$ in $p(x_t | x_{t+1})$.

conditional probability as a function of $x_{t-\Delta t}$. Now:

$$\log p(x_{t-\Delta t}|x_t) = \log p(x_t|x_{t-\Delta t}) + \log p_{t-\Delta t}(x_{t-\Delta t}) - \cancel{\log p_t(x_t)}$$ Drop constants involving only x_t.

$$= \log p(x_t|x_{t-\Delta t}) + \log p_t(x_{t-\Delta t}) + \mathcal{O}(\Delta t)$$ Since $p_{t-\Delta t}(\cdot) = p_t(\cdot) + \Delta t \frac{\partial}{\partial t} p_t(\cdot)$.

$$= -\frac{1}{2\sigma_q^2 \Delta t}||x_{t-\Delta t} - x_t||_2^2 + \log p_t(x_{t-\Delta t})$$ Definition of $\log p(x_t|x_{t-\Delta t})$.

$$= -\frac{1}{2\sigma_q^2 \Delta t}||x_{t-\Delta t} - x_t||_2^2 + \cancel{\log p_t(x_t)} + \langle \nabla_x \log p_t(x_t), (x_{t-\Delta t} - x_t)\rangle + \mathcal{O}(\Delta t)$$ Taylor expand around x_t and drop constants.

$$= -\frac{1}{2\sigma_q^2 \Delta t}\left(||x_{t-\Delta t} - x_t||_2^2 - 2\sigma_q^2 \Delta t \langle \nabla_x \log p_t(x_t), (x_{t-\Delta t} - x_t)\rangle\right)$$

$$= -\frac{1}{2\sigma_q^2 \Delta t}||x_{t-\Delta t} - x_t - \sigma_q^2 \Delta t \ \nabla_x \log p_t(x_t)||_2^2 + C$$ Complete the square in $(x_{t-\Delta t} - x_t)$, and drop constant C involving only x_t.

$$= -\frac{1}{2\sigma_q^2 \Delta t}||x_{t-\Delta t} - \mu||_2^2$$ For $\mu := x_t + (\sigma_q^2 \Delta t)\nabla_x \log p_t(x_t)$.

This is identical, up to additive factors, to the log-density of a Normal distribution with mean μ and variance $\sigma_q^2 \Delta t$. Therefore,

$$p(x_{t-\Delta t} \mid x_t) \approx \mathcal{N}(x_{t-\Delta t};\ \mu, \sigma_q^2 \Delta t). \tag{20}$$

□

Reflecting on this derivation, the main idea was that for small enough Δt, the Bayes-rule expansion of the reverse process $p(x_{t-\Delta t} \mid x_t)$ is dominated by the term $p(x_t \mid x_{t-\Delta t})$, from the forward process. This is intuitively why the reverse process and the forward process have the same functional form (both are Gaussian here)[17].

[17] This general relationship between forward and reverse processes holds somewhat more generally than just Gaussian diffusion; see e.g. the discussion in Sohl-Dickstein et al. [2015].

Technical Details [Optional]. The meticulous reader may notice that Claim 1 is not obviously sufficient to imply correctness of the entire DDPM algorithm. The issue is: as we scale down Δt, the error in our per-step approximation (Equation 16) decreases, but the number of total steps required increases. So if the per-step error does not decrease fast enough (as a function of Δt), then these errors could accumulate to a non-negligible error by the final step. Thus, we need to quantify how fast the per-step error decays. Lemma 1 below is one way of quantifying this: it states that if the step-size (i.e. variance of the per-step noise) is σ^2, then the KL error of the per-step Gaussian approximation is $\mathcal{O}(\sigma^4)$. This decay rate is fast enough, because the number of steps only grows as[18] $\Omega(1/\sigma^2)$.

[18] The chain rule for KL implies that we can add up these per-step errors: the approximation error for the final sample is bounded by the sum of all the per-step errors.

Lemma 1. *Let $p(x)$ be an arbitrary density over $\mathbb{R}$, with bounded 1st to 4th order derivatives. Consider the joint distribution (x_0, x_1), where $x_0 \sim p$ and $x_1 \sim x_0 + \mathcal{N}(0, \sigma^2)$. Then, for any conditioning $z \in \mathbb{R}$, we have*

$$\mathrm{KL}\left(\mathcal{N}(\mu_z, \sigma^2) \ ||\ p_{x_0|x_1}(\cdot \mid x_1 = z)\right) \leq O(\sigma^4), \tag{21}$$

where

$$\mu_z := z + \sigma^2 \nabla \log p(z). \tag{22}$$

It is possible to prove Lemma 1 by doing essentially a careful Taylor expansion; we include the full proof in Appendix B.1.

2.2 *Algorithms*

Pseudocode listings 1 and 2 give the explicit DDPM train loss and sampling code. To train[19] the network f_θ, we must minimize the expected loss L_θ output by Pseudocode 1, typically by backpropagation.

Pseudocode 3 describes the closely-related *DDIM* sampler, which will be discussed later in Section 3.

[19] Note that the training procedure optimizes f_θ for all timesteps t simultaneously, by sampling $t \in [0,1]$ uniformly in Line 2.

Pseudocode 1: DDPM train loss

Input: Neural network f_θ; Sample-access to target distribution p.
Data: Terminal variance σ_q; step-size Δt.
Output: Stochastic loss L_θ

1 $x_0 \leftarrow \text{Sample}(p)$
2 $t \leftarrow \text{Unif}[0,1]$
3 $x_t \leftarrow x_0 + \mathcal{N}(0, \sigma_q^2 t)$
4 $x_{t+\Delta t} \leftarrow x_t + \mathcal{N}(0, \sigma_q^2 \Delta t)$
5 $L \leftarrow \|f_\theta(x_{t+\Delta t}, t+\Delta t) - x_t\|_2^2$
6 **return** L_θ

Pseudocode 2: DDPM sampling (Code for Algorithm 1)

Input: Trained model f_θ.
Data: Terminal variance σ_q; step-size Δt.
Output: x_0

1 $x_1 \leftarrow \mathcal{N}(0, \sigma_q^2)$
2 **for** $t = 1,\ (1-\Delta t), (1-2\Delta t), \ldots, \Delta t$ **do**
3 $\quad \eta \leftarrow \mathcal{N}(0, \sigma_q^2 \Delta t)$
4 $\quad x_{t-\Delta t} \leftarrow f_\theta(x_t, t) + \eta$
5 **end**
6 **return** x_0

Pseudocode 3: DDIM sampling (Code for Algorithm 2)

Input: Trained model f_θ
Data: Terminal variance σ_q; step-size Δt.
Output: x_0

1 $x_1 \leftarrow \mathcal{N}(0, \sigma_q^2)$
2 **for** $t = 1,\ (1-\Delta t), (1-2\Delta t), \ldots, \Delta t, 0$ **do**
3 $\quad \lambda \leftarrow \frac{\sqrt{t}}{\sqrt{t-\Delta t}+\sqrt{t}}$
4 $\quad x_{t-\Delta t} \leftarrow x_t + \lambda(f_\theta(x_t, t) - x_t)$
5 **end**
6 **return** x_0

2.3 *Variance Reduction: Predicting x_0*

Thus far, our diffusion models have been trained to predict $\mathbb{E}[x_{t-\Delta t} \mid x_t]$: this is what Algorithm 1 requires, and what the training procedure of Pseudocode 1 produces. However, many practical

diffusion implementations actually train to predict $\mathbb{E}[x_0 \mid x_t]$, i.e. to predict the expectation of the initial point x_0 instead of the previous point $x_{t-\Delta t}$. This difference turns out to be just a *variance reduction* trick, which estimates the same quantity in expectation. Formally, the two quantities can be related as follows:

Claim 2. *For the Gaussian diffusion setting of Section 1.3 , we have:*

$$\mathbb{E}[(x_{t-\Delta t} - x_t) \mid x_t] = \frac{\Delta t}{t}\,\mathbb{E}[(x_0 - x_t) \mid x_t]. \tag{23}$$

Or equivalently:

$$\mathbb{E}[x_{t-\Delta t} \mid x_t] = \left(\frac{\Delta t}{t}\right)\mathbb{E}[x_0 \mid x_t] + \left(1 - \frac{\Delta t}{t}\right)x_t. \tag{24}$$

This claim implies that if we want to estimate $\mathbb{E}[x_{t-\Delta t} \mid x_t]$, we can instead estimate $\mathbb{E}[x_0 \mid x_t]$ and then then essentially divide by $(t/\Delta t)$, which is the number of steps taken thus far. The variance-reduced versions of the DDPM training and sampling algorithms do exactly this; we include them in Appendix B.9.

The intuition behind Claim 2 is illustrated in Figure 3: first, observe that predicting $x_{t-\Delta t}$ given x_t is equivalent to predicting the last noise step, which is $\eta_{t-\Delta t} = (x_t - x_{t-\Delta t})$ in the forward process of Equation (11). But, if we are only given the final x_t, then all of the previous noise steps $\{\eta_i\}_{i<t}$ intuitively "look the same"— we cannot distinguish between noise that was added at the last step from noise that was added at the 5th step, for example. By this symmetry, we can conclude that all of the individual noise steps are distributed *identically* (though not independently) given x_t. Thus, instead of estimating a single noise step, we can equivalently estimate the *average of all prior noise steps*, which has much lower variance. There are $(t/\Delta t)$ elapsed noise steps by time t, so we divide the total noise by this quantity in Equation 23 to compute the average. See Appendix B.8 for a formal proof.

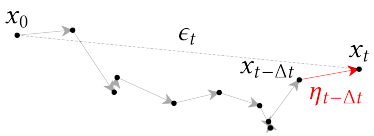

Figure 3: The intuition behind Claim 2. Given x_t, the final noise step $\eta_{t-\Delta t}$ is distributed identically as all other noise steps, intuitively because we only know the sum $x_t = x_0 + \sum_i \eta_i$.

WORD OF WARNING: Diffusion models should always be trained to estimate *expectations*. In particular, when we train a model to predict $\mathbb{E}[x_0 \mid x_t]$, we should *not* think of this as trying to learn "how to sample from the distribution $p(x_0 \mid x_t)$". For example, if we are training an image diffusion model, then the optimal model will output $\mathbb{E}[x_0 \mid x_t]$ which will look like a blurry mix of images (e.g. Figure 1b in Karras et al. [2022])— it will not look like an actual image sample. It is good to keep in mind that when diffusion papers colloquially discuss models "predicting x_0", they do not mean producing something that looks like an actual sample of x_0.

2.4 *Diffusions as SDEs [Optional]*

In this section[20], we connect the discrete-time processes we have discussed so far to stochastic differential equations (SDEs). In the continuous limit, as $\Delta t \to 0$, our discrete diffusion process turns into a stochastic differential equation. SDEs can also represent many other diffusion variants (corresponding to different drift and diffusion terms), offering flexibility in design choices, like scaling and noise-scheduling. The SDE perspective is powerful because existing theory provides a general closed-form solution for the time-reversed SDE. Discretization of the reverse-time SDE for our particular diffusion immediately yields the sampler we derived in this section, but reverse-time SDEs for other diffusion variants are also available *automatically* (and can then be solved with any off-the-shelf or custom SDE solver), enabling better training and sampling strategies as we will discuss further in Section 5. Though we mention these connections only briefly here, the SDE perspective has had significant impact on the field. For a more detailed discussion, we recommend Yang Song's blog post [Song, 2021].

[20] Sections marked "[Optional]" are advanced material, and can be skipped on first read. None of the main sections depend on Optional material.

The Limiting SDE

Recall our discrete update rule:

$$x_{t+\Delta t} = x_t + \sigma_q \sqrt{\Delta t}\xi, \quad \xi \sim \mathcal{N}(0,1).$$

In this limit as $\Delta t \to 0$, this corresponds to a zero-drift SDE:

$$dx = \sigma_q dw, \tag{25}$$

where w is a Brownian motion. A Brownian motion is a stochastic process with i.i.d. Gaussian increments whose variance scales with Δt.[21] *Very heuristically*, we can think of $dw \sim \lim_{\Delta t \to 0} \sqrt{\Delta t}\mathcal{N}(0,1)$, and thus "derive" (25) by

$$dx = \lim_{\Delta t \to 0} (x_{t+\Delta t} - x_t) = \sigma_q \lim_{\Delta t \to 0} \sqrt{\Delta t}\xi = \sigma_q dw.$$

[21] See Eldan [2024] for a high-level overview of Brownian motions and Itô's formula. See also Evans [2012] for a gentle introductory textbook, and Kloeden and Platen [2011] for numerical methods.

More generally, different variants of diffusion are equivalent to SDEs with different choices of drift and diffusion terms:

$$dx = f(x,t)dt + g(t)dw. \tag{26}$$

The SDE (25) simply has $f = 0$ and $g = \sigma_q$. This formulation encompasses many other possibilities, though, corresponding to different choices of f, g in the SDE. As we will revisit in Section 5, this flexibility is important for developing effective algorithms. Two important

choices made in practice are tuning the noise schedule and scaling x_t; together these can help to control the variance of x_t, and control how much we focus on different noise levels. Adopting a flexible noise schedule $\{\sigma_t\}$ in place of the fixed schedule $\sigma_t \equiv \sigma_q\sqrt{t}$ corresponds to the SDE [Song et al., 2020]

$$x_t \sim \mathcal{N}(x_0, \sigma_t^2) \iff x_t = x_{t-\Delta t} + \sqrt{\sigma_t^2 - \sigma_{t-\Delta t}^2}z_{t-\Delta t} \iff dx = \sqrt{\frac{d}{dt}\sigma^2(t)}dw.$$

If we also wish to scale each x_t by a factor $s(t)$, Karras et al. [2022] show that this corresponds to the SDE [22]

$$x_t \sim \mathcal{N}(s(t)x_0, s(t)^2\sigma(t)^2) \iff f(x) = \frac{\dot{s}(t)}{s(t)}x, \quad g(t) = s(t)\sqrt{2\dot{\sigma}(t)\sigma(t)}.$$

These are only a few examples of the rich and useful design space enabled by the flexible SDE (26).

[22] As a sketch of how f arises, let's ignore the noise and note that:

$$\begin{aligned} & x_t = s(t)x_0 \\ \iff & x_{t+\Delta t} = \frac{s(t+\Delta t)}{s(t)}x_t \\ & = x_t + \frac{s(t) - s(t+\Delta t)}{s(t)}x_t \\ \iff & dx/dt = \frac{\dot{s}}{s}x \end{aligned}$$

Reverse-Time SDE

The *time-reversal* of an SDE runs the process *backward in time*. Reverse-time SDEs are the continuous-time analog of samplers like DDPM. A deep result due to Anderson [1982] (and nicely re-derived in Winkler [2021]) states that the time-reversal of SDE (26) is given by:

$$dx = \big(f(x,t) - g(t)^2\nabla_x \log p_t(x)\big)dt + g(t)d\overline{w} \tag{27}$$

That is, SDE (27) tells us how to run any SDE of the form (26) backward in time! This means that we don't have to re-derive the reversal in each case, and we can choose any SDE solver to yield a practical sampler. But nothing is free: we sill cannot use (27) directly to sample backward, since the term $\nabla_x \log p_t(x)$ – which is in fact the *score* that previously appeared in equation 18 – is unknown in general, since it depends on p_t. However, if we can *learn* the score, then we can solve the reverse SDE. This is analogous to discrete diffusion, where the *forward* process is easy to model (it just adds noise), while the *reverse* process must be learned.

Let us take a moment to discuss the score, $\nabla_x \log p_t(x)$, which plays a central role. Intuitively, since the score "points toward higher probability", it helps to reverse the diffusion process, which "flattens out" the probability as it runs forward. The score is also related to the conditional expectation of x_0 given x_t. Recall that in the discrete case

$$\sigma_q^2\Delta t\nabla \log p_t(x_t) = \mathbb{E}[x_{t-\Delta t} - x_t \mid x_t] = \frac{\Delta t}{t}\mathbb{E}[x_0 - x_t \mid x_t],$$

(by equations 18, 23).

Similarly, in the continuous case we have [23]

$$\sigma_q^2 \nabla \log p_t(x_t) = \frac{1}{t}\,\mathbb{E}[x_0 - x_t \mid x_t]. \tag{28}$$

Returning to the reverse SDE, we can show that its discretization yields the DDPM sampler of Claim 1 as a special case. The reversal of the simple SDE (25) is:

$$\begin{aligned} dx &= -\sigma_q^2 \nabla_x \log p_t(x) dt + \sigma_q d\overline{w} && (29)\\ &= -\frac{1}{t}\,\mathbb{E}[x_0 - x_t \mid x_t] dt + \sigma_q d\overline{w} && (30) \end{aligned}$$

The discretization is

$$\begin{aligned} x_t - x_{t-\Delta t} &= -\frac{\Delta t}{t}\,\mathbb{E}[x_0 - x_t \mid x_t] + \mathcal{N}(0, \sigma_q^2 \Delta t) && (31)\\ &= -\,\mathbb{E}[x_{t-\Delta t} - x_t \mid x_t] + \mathcal{N}(0, \sigma_q^2 \Delta t) && \text{(by Eqn. 23)}\\ \implies x_{t-\Delta t} &= \mathbb{E}[x_{t-\Delta t} \mid x_t] + \mathcal{N}(0, \sigma_q^2 \Delta t) && (32) \end{aligned}$$

which is exactly the stochastic (DDPM) sampler derived in Claim 1.

[23] We can see this directly by applying Tweedie's formula, which states:

$\mathbb{E}[\mu_z|z] = z + \sigma_z^2 \nabla \log p(z)$ for $z \sim \mathcal{N}(\mu_z, \sigma_z^2)$.

Since $x_t \sim \mathcal{N}(x_0, t\sigma_q^2)$, Tweedie with $z \equiv x_t$, $\mu_z \equiv x_0$ gives:

$$\mathbb{E}[x_0|x_t] = x_t + t\sigma_q^2 \nabla \log p(x_t).$$

3 Deterministic Sampling: DDIM

We will now show a *deterministic* reverse sampler for Gaussian diffusion— which appears similar to the stochastic sampler of the previous section, but is conceptually quite different. This sampler is equivalent to the DDIM[24] update of Song et al. [2021], adapted to in our simplified setting.

We consider the same Gaussian diffusion setup as the previous section, with the joint distribution $(x_0, x_{\Delta t}, \ldots, x_1)$ and conditional expectation function $\mu_t(z) := \mathbb{E}[x_t \mid x_{t+\Delta t} = z]$. The reverse sampler is defined below, and listed explicitly in Pseudocode 3.

Algorithm 2: Deterministic Reverse Sampler (DDIM-like)

For input sample x_t, and step index t, output:

$$\widehat{x}_{t-\Delta t} \leftarrow x_t + \lambda(\mu_{t-\Delta t}(x_t) - x_t) \tag{33}$$

where $\lambda := \left(\frac{\sigma_t}{\sigma_{t-\Delta t}+\sigma_t}\right)$ and $\sigma_t \equiv \sigma_q\sqrt{t}$ from Equation (12).

How do we show that this defines a valid reverse sampler? Since Algorithm 2 is *deterministic*, it does not make sense to argue that it *samples* from $p(x_{t-\Delta t} \mid x_t)$, as we argued for the DDPM-like stochastic sampler. Instead, we will directly show that Equation (33) implements a valid *transport map* between the marginal distributions p_t and $p_{t-\Delta t}$. That is, if we let F_t be the update of Equation (33):

$$F_t(z) := z + \lambda(\mu_{t-\Delta t}(z) - z) \tag{34}$$

$$= z + \lambda(\mathbb{E}[x_{t-\Delta t} \mid x_t = z] - z) \tag{35}$$

then we want to show that[25]

$$F_t \sharp p_t \approx p_{t-\Delta t}. \tag{36}$$

Proof overview: The usual way to prove this is to use tools from stochastic calculus, but we'll present an elementary derivation. Our strategy will be to first show that Algorithm 2 is correct in the simplest case of a point-mass distribution, and then lift this result to full distributions by marginalizing appropriately. For the experts, this is similar to "flow-matching" proofs.

3.1 Case 1: Single Point

Let's first understand the simple case where the target distribution p_0 is a single point mass in $\mathbb{R}^d$. Without loss of generality[26], we can assume the point is at $x_0 = 0$. Is Algorithm 2 correct in this case?

[24] DDIM stands for Denoising Diffusion Implicit Models, which reflects a perspective used in the original derivation of Song et al. [2021]. Our derivation follows a different perspective, and the "implicit" aspect will not be important to us.

[25] The notation $F\sharp p$ means the distribution of $\{F(x)\}_{x\sim p}$. This is called the *pushforward* of p by the function F.

[26] Because we can just "shift" our coordinates to make it so. Formally, our entire setup including Equation 35 is translation-symmetric.

To reason about correctness, we want to consider the distributions of x_t and $x_{t-\Delta t}$ for arbitrary step t. According to the diffusion forward process (Equation 11), at time t the relevant random variables are[27]

$$\begin{aligned} x_0 &= 0 \quad \text{(deterministically)} \\ x_{t-\Delta t} &\sim \mathcal{N}(x_0, \sigma^2_{t-\Delta t}) \\ x_t &\sim \mathcal{N}(x_{t-\Delta t}, \sigma^2_t - \sigma^2_{t-\Delta t}). \end{aligned}$$

[27] We omit the Identity matrix in these covariances for notational simplicity. The reader may assume dimension $d = 1$ without loss of generality.

The marginal distribution of $x_{t-\Delta t}$ is $p_{t-\Delta t} = \mathcal{N}(0, \sigma^2_{t-1})$, and the marginal distribution of x_t is $p_t = \mathcal{N}(0, \sigma^2_t)$.

Let us first find *some* deterministic function $G_t : \mathbb{R}^d \to \mathbb{R}^d$, such that $G_t \sharp p_t = p_{t-\Delta t}$. There are many possible functions which will work[28], but this is the obvious one:

$$G_t(z) := \left(\frac{\sigma_{t-\Delta t}}{\sigma_t}\right) z. \tag{37}$$

[28] For example, we can always add a rotation around the origin to any valid map.

The function G_t above simply re-scales the Gaussian distribution of p_t, to match variance of the Gaussian distribution $p_{t-\Delta t}$. It turns out this G_t is exactly equivalent to the step F_t taken by Algorithm 2, which we will now show.

Claim 3. *When the target distribution is a point mass $p_0 = \delta_0$, then update F_t (as defined in Equation 35) is equivalent to the scaling G_t (as defined in Equation 37):*

$$F_t \equiv G_t. \tag{38}$$

Thus Algorithm 2 defines a reverse sampler for target distribution $p_0 = \delta_0$.

Proof. To apply F_t, we need to compute $\mathbb{E}[x_{t-\Delta t} \mid x_t]$ for our simple distribution. Since $(x_{t-\Delta t}, x_t)$ are jointly Gaussian, this is[29]

$$\mathbb{E}[x_{t-\Delta t} \mid x_t] = \left(\frac{\sigma^2_{t-\Delta t}}{\sigma^2_t}\right) x_t. \tag{39}$$

[29] Recall the conditional expectation of two jointly Gaussian random variables (X, Y) is $\mathbb{E}[X \mid Y = y] = \mu_X + \Sigma_{XY}\Sigma^{-1}_{YY}(y - \mu_Y)$, where μ_X, μ_Y are the respective means, and Σ_{XY}, Σ_{YY} the cross-covariance of (X, Y) and covariance of Y. Since $X = x_{t-\Delta t}$ and $Y = x_t$ are centered at 0, we have $\mu_X = \mu_Y = 0$. For the covariance term, since $x_t = x_{t-\Delta t} + \eta$ we have $\Sigma_{XY} = \mathbb{E}[x_t x^T_{t-\Delta t}] = \mathbb{E}[x_{t-\Delta t} x^T_{t-\Delta t}] = \sigma^2_{t-\Delta t} I_d$. Similarly, $\Sigma_{YY} = \mathbb{E}[x_t x^T_t] = \sigma^2_t I_d$.

The rest is algebra:

$$\begin{aligned} F_t(x_t) &:= x_t + \lambda(\mathbb{E}[x_{t-\Delta t} \mid x_t] - x_t) && \text{by definition of } F_t \\ &= x_t + \left(\frac{\sigma_t}{\sigma_{t-\Delta t} + \sigma_t}\right)(\mathbb{E}[x_{t-\Delta t} \mid x_t] - x_t) && \text{by definition of } \lambda \\ &= x_t + \left(\frac{\sigma_t}{\sigma_{t-\Delta t} + \sigma_t}\right)\left(\frac{\sigma^2_{t-\Delta t}}{\sigma^2_t} - 1\right) x_t && \text{by Equation (39)} \\ &= \left(\frac{\sigma_{t-\Delta t}}{\sigma_t}\right) x_t \\ &= G_t(x_t). \end{aligned}$$

We therefore conclude that Algorithm 2 is a correct reverse sampler, since it is equivalent to G_t, and G_t is valid. □

The correctness of Algorithm 2 still holds[30] if x_0 is an arbitrary point instead of $x_0 = 0$, since everything is transitionally symmetric.

[30] See Claim 5 in Appendix B.3 for an explicit statement.

3.2 *Velocity Fields and Gases*

Before we move on, it will be helpful to think of the DDIM update as equivalent to a velocity field, which moves points at time t to their positions at time $(t - \Delta t)$. Specifically, define the vector field

$$v_t(x_t) := \frac{\lambda}{\Delta t}(\mathbb{E}[x_{t-\Delta t} \mid x_t] - x_t). \tag{40}$$

Then the DDIM update algorithm of Equation (33) can be written as:

$$\begin{aligned} \widehat{x}_{t-\Delta t} &:= x_t + \lambda(\mu_{t-\Delta t}(x_t) - x_t) && \text{from Equation (33)} \\ &= x_t + v_t(x_t)\Delta t. \end{aligned} \tag{41}$$

The physical intuition for v_t is: imagine a gas of non-interacting particles, with density field given by p_t. Then, suppose a particle at position z moves in the direction $v_t(z)$. The resulting gas will have density field $p_{t-\Delta t}$. We write this process as

$$p_t \xrightarrow{v_t} p_{t-\Delta t}. \tag{42}$$

In the limit of small stepsize Δt, speaking informally, we can think of v_t as a *velocity field* — which specifies the instantaneous velocity of particles moving according to the DDIM algorithm.

As a concrete example, if the target distribution $p_0 = \delta_{x_0}$, as in Section 3.1, then the velocity field of DDIM is $v_t(x_t) = \left(\frac{\sigma_t - \sigma_{t-\Delta t}}{\sigma_t}\right)(x_0 - x_t)/\Delta t$ which is a vector field that always points towards the initial point x_0 (see Figure 4).

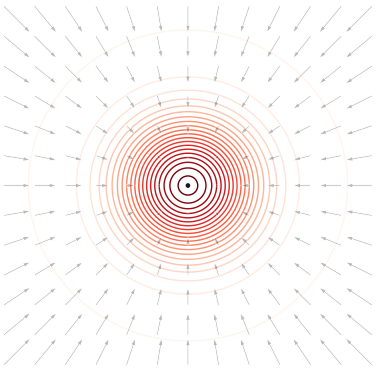

Figure 4: Velocity field v_t when $p_0 = \delta_{x_0}$, overlaid on the Gaussian distribution p_t.

3.3 *Case 2: Two Points*

Now let us show Algorithm 2 is correct when the target distribution is a mixture of two points:

$$p_0 := \frac{1}{2}\delta_a + \frac{1}{2}\delta_b, \tag{43}$$

for some $a, b \in \mathbb{R}^d$. According to the diffusion forward process, the distribution at time t will be a mixture of Gaussians[31]:

$$p_t := \frac{1}{2}\mathcal{N}(a, \sigma_t^2) + \frac{1}{2}\mathcal{N}(b, \sigma_t^2). \tag{44}$$

We want to show that with these distributions p_t, the DDIM velocity field v_t (of Equation 40) transports $p_t \xrightarrow{v_t} p_{t-\Delta t}$.

Let us first try to construct *some* velocity field v_t^* such that $p_t \xrightarrow{v_t^*} p_{t-\Delta t}$. From our result in Section 3.1 — the fact that DDIM update works for single points — we already know velocity fields

[31] Linearity of the forward process (with respect to p_0) was important here. That is, roughly speaking, diffusing a distribution is equivalent to diffusing each individual point in that distribution independently; the points don't interact.

which transport each mixture component $\{a, b\}$ individually. That is, we know the velocity field $v_t^{[a]}$ defined as

$$v_t^{[a]}(x_t) := \lambda \mathop{\mathbb{E}}_{x_0 \sim \delta_a} [x_{t-\Delta t} - x_t \mid x_t] \tag{45}$$

transports[32]

$$\mathcal{N}(a, \sigma_t^2) \xrightarrow{v_t^{[a]}} \mathcal{N}(a, \sigma_{t-\Delta t}^2), \tag{46}$$

and similarly for $v_t^{[b]}$.

We now want some way of combining these two velocity fields into a single velocity v_t^*, which transports the mixture:

$$\underbrace{\left(\frac{1}{2}\mathcal{N}(a, \sigma_t^2) + \frac{1}{2}\mathcal{N}(b, \sigma_t^2)\right)}_{p_t} \xrightarrow{v_t^*} \underbrace{\left(\frac{1}{2}\mathcal{N}(a, \sigma_{t-\Delta t}^2) + \frac{1}{2}\mathcal{N}(b, \sigma_{t-\Delta t}^2)\right)}_{p_{t-\Delta t}} \tag{47}$$

We may be tempted to just take the average velocity field ($v_t^* = 0.5 v_t^{[a]} + 0.5 v_t^{[b]}$), but this is incorrect. The correct combined velocity v_t^* is a *weighted*-average of the individual velocity fields, weighted by their corresponding density fields[33].

$$v_t^*(x_t) = \frac{v_t^{[a]}(x_t) \cdot p(x_t \mid x_0 = a) + v_t^{[b]}(x_t) \cdot p(x_t \mid x_0 = b)}{p(x_t \mid x_0 = a) + p(x_t \mid x_0 = b)} \tag{48}$$

$$= v_t^{[a]}(x_t) \cdot p(x_0 = a \mid x_t) + v_t^{[b]}(x_t) \cdot p(x_0 = b \mid x_t). \tag{49}$$

Explicitly, the weight for $v_t^{[a]}$ at a point x_t is the probability that x_t was generated from initial point $x_0 = a$, rather than $x_0 = b$.

To be intuitively convinced of this[34], consider the corresponding question about gasses illustrated in Figure 5. Suppose we have two overlapping gases, a red gas with density $\mathcal{N}(a, \sigma^2)$ and velocity $v_t^{[a]}$, and a blue gas with density $\mathcal{N}(b, \sigma^2)$ and velocity $v_t^{[b]}$. We want to know, what is the effective velocity of the combined gas (as if we saw only in grayscale)? We should clearly take a *weighted*-average of the individual gas velocities, weighted by their respective densities — just as in Equation (49).

We have now solved the main subproblem of this section: we have found *one particular* vector field v_t^* which transports p_t to $p_{t-\Delta t}$, for our two-point distribution p_0. It remains to show that this v_t^* is equivalent to the velocity field of Algorithm 2 (v_t from Equation 40).

[32] Pay careful attention to which distributions we take expectations over! The expectation in Equation (45) is w.r.t. the single-point distribution δ_a, but our definition of the DDIM algorithm, and its vector field in Equation (40), are always w.r.t. the target distribution. In our case, the target distribution is p_0 of Equation (43).

[33] Note that we can write the density $\mathcal{N}(x_t; a, \sigma_t^2)$ as $p(x_t \mid x_0 = a)$.

[34] The time step must be small enough for this analogy to hold, so the DDIM updates are essentially infinitesimal steps. Otherwise, if the step size is large, it may not be possible to combine the two transport maps with "local" (i.e. pointwise) operations alone.

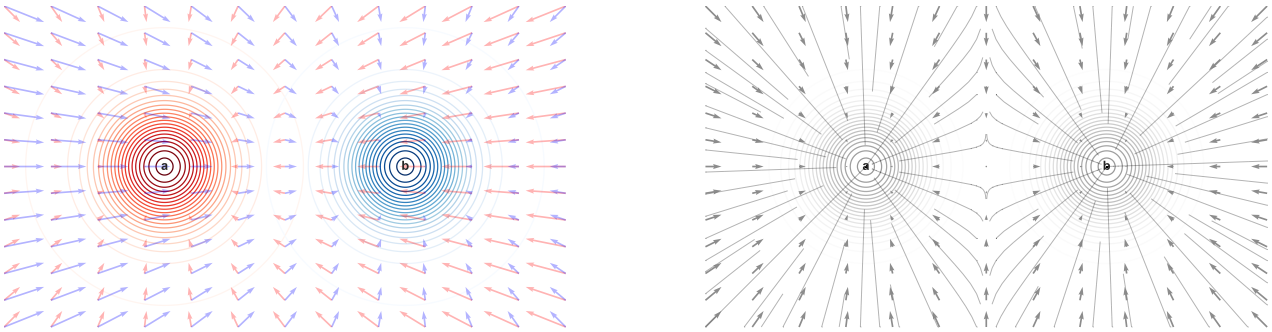

Figure 5: Illustration of combining the velocity fields of two gasses. Left: The density and velocity fields of two independent gases (in red and blue). Right: The effective density and velocity field of the combined gas, including streamlines.

To show this, first notice that the individual vector field $v_t^{[a]}$ can be written as a conditional expectation. Using the definition in Equation (45)[35],

$$v_t^{[a]}(x_t) = \lambda \mathop{\mathbb{E}}_{x_0 \sim \delta_a} [x_{t-\Delta t} - x_t \mid x_t] \tag{50}$$

$$= \lambda \mathop{\mathbb{E}}_{x_0 \sim 1/2\delta_a + 1/2\delta_b} [x_{t-\Delta t} - x_t \mid x_0 = a, x_t]. \tag{51}$$

[35] We add conditioning $x_0 = a$, because we want to take expectations w.r.t the two-point mixture distribution, not the single-point distribution.

Now the entire vector field v_t^* can be written as a conditional expectation:

$$v_t^*(x_t) = v_t^{[a]}(x_t) \cdot p(x_0 = a \mid x_t) + v_t^{[b]}(x_t) \cdot p(x_0 = b \mid x_t) \tag{52}$$

$$= \lambda\, \mathbb{E}[x_{t-\Delta t} - x_t \mid x_0 = a, x_t] \cdot p(x_0 = a \mid x_t) \tag{53}$$

$$+ \lambda\, \mathbb{E}[x_{t-\Delta t} - x_t \mid x_0 = b, x_t] \cdot p(x_0 = b \mid x_t) \tag{54}$$

$$= \lambda\, \mathbb{E}\left[x_{t-\Delta t} - x_t \mid x_t\right] \tag{55}$$

$$= v_t(x_t) \qquad \text{(from Equation 40)}$$

where all expectations are w.r.t. the distribution $x_0 \sim 1/2\delta_a + 1/2\delta_b$. Thus, the combined velocity field v_t^* is exactly the velocity field v_t given by the updates of Algorithm 2 — so Algorithm 2 is a correct reverse sampler for our two-point mixture distribution.

3.4 *Case 3: Arbitrary Distributions*

Now that we know how to handle two points, we can generalize this idea to arbitrary distributions of x_0. We will not go into details here, because the general proof will be subsumed by the subsequent section.

It turns out that our overall proof strategy for Algorithm 2 can be generalized significantly to other types of diffusions, without much

work. This yields the idea of flow matching, which we will see in the following section. Once we develop the machinery of flows, it is actually straightforward to derive DDIM directly from the simple single-point scaling algorithm of Equation (37): see Appendix B.5.

3.5 *The Probability Flow ODE [Optional]*

Finally, we generalize our discrete-time deterministic sampler to an ordinary differential equation (ODE) called the *probability flow ODE* [Song et al., 2020]. The following section builds on our discussion of SDEs as the continuous limit of diffusion in section 2.4. Just as the reverse-time SDEs of section 2.4 offered a flexible continuous-time generalization of discrete stochastic samplers, so we will see that discrete deterministic samplers generalize to ODEs. The ODE formulation offers both a useful theoretical lens through which to view diffusion, as well as practical advantages, like the opportunity to choose from a variety of off-the-shelf and custom ODE solvers to improve sampling (like the popular DPM++ method, as discussed in chapter 5).

Recall the general SDE (26) from section 2.4:

$$dx = f(x,t)dt + g(t)dw.$$

Song et al. [2020] showed that is possible to convert this SDE into a *deterministic* equivalent called the probability flow ODE (PF-ODE): [36]

$$\frac{dx}{dt} = \tilde{f}(x,t), \quad \text{where } \tilde{f}(x,t) = f(x,t) - \frac{1}{2}g(t)^2\nabla_x \log p_t(x) \tag{56}$$

SDE (26) and ODE (56) are equivalent in the sense that trajectories obtained by solving the PF-ODE have the same marginal distributions as the SDE trajectories at every point in time[37]. However, note that the score appears here again, as it did in the reverse SDE (27); just as for the reverse SDE, we must learn the score to make the ODE (56) practically useful.

Just as DDPM was a (discretized) special-case of the reverse-time SDE (27), so DDIM can be seen as a (discretized) special case of the PF-ODE (56). Recall from section 2.4 that the simple diffusion we have been studying corresponds to the SDE (25) with $f = 0$ and $g = \sigma_q$. The corresponding ODE is

$$\frac{dx}{dt} = -\frac{1}{2}\sigma_q^2\nabla_x \log p_t(x) \tag{57}$$

$$= -\frac{1}{2t}\mathbb{E}[x_0 - x_t \mid x_t] \quad \text{(by eq. 28)} \tag{58}$$

[36] A proof sketch is in appendix B.2. It involves rewriting the SDE noise term as the deterministic score (recall the connection between noise and score in equation (18)). Although it is deterministic, the score is unknown since it depends on p_t.

[37] To use a gas analogy: the SDE describes the (Brownian) motion of individual particles in a gas, while the PF-ODE describes the streamlines of the gas's velocity field. That is, the PF-ODE describes the motion of a "test particle" being transported by the gas— like a feather in the wind.

Reversing and discretizing yields

$$\begin{aligned} x_{t-\Delta t} &= x_t + \frac{\Delta t}{2t}\,\mathbb{E}[x_0 - x_t \mid x_t] \\ &= x_t + \frac{1}{2}(\mathbb{E}[x_{t-\Delta t} \mid x_t] - x_t) \quad \text{(by eq. 23)}. \end{aligned}$$

Noting that $\lim_{\Delta t \to 0} \left(\frac{\sigma_t}{\sigma_{t-\Delta t}+\sigma_t}\right) = \frac{1}{2}$, we recover the deterministic (DDIM) sampler (33).

3.6 *Discussion: DDPM vs DDIM*

The two reverse samplers defined above (DDPM and DDIM) are conceptually significantly different: one is deterministic, and the other stochastic. To review, these samplers use the following strategies:

1. DDPM ideally implements a stochastic map F_t, such that the output $F_t(x_t)$ is, pointwise, a sample from the conditional distribution $p(x_{t-\Delta t} \mid x_t)$.
2. DDIM ideally implements a deterministic map F_t, such that the output $F_t(x_t)$ is *marginally distributed* as $p_{t-\Delta t}$. That is, $F_t \sharp p_t = p_{t-\Delta t}$.

Although they both happen to take steps in the same direction[38] (given the same input x_t), the two algorithms end up evolving very differently. To see this, let's consider how each sampler ideally behaves, when started from the same initial point x_1 and iterated to completion.

[38] Steps proportional to $(\mu_{t-\Delta t}(x_t) - x_t)$.

DDPM will ideally produce a sample from $p(x_0 \mid x_1)$. If the forward process mixes sufficiently (i.e. for large σ_q in our setup), then the final point x_1 will be nearly *independent* from the initial point. Thus $p(x_0 \mid x_1) \approx p(x_0)$, so the distribution output by the ideal DDPM will not depend at all[39] on the starting point x_1. In contrast, DDIM is deterministic, so it will always produce a fixed value for a given x_1, and thus will depend very strongly on x_1.

[39] Actual DDPMs may have a small dependency on the initial point x_1, because they do not mix perfectly (i.e. the final distribution p_1 is not perfectly Gaussian). Randomizing the initial point may thus help with sample diversity in practice.

The picture to have in mind is, DDIM defines a deterministic map $\mathbb{R}^d \to \mathbb{R}^d$, taking samples from a Gaussian distribution to our target distribution. At this level, the DDIM map may sound similar to other generative models — after all, GANs and Normalizing Flows also define maps from Gaussian noise to the true distribution. What is special about the DDIM map is, it is not allowed to be arbitrary: the target distribution p^* exactly determines the ideal DDIM map (which we train models to emulate). This map is "nice"; for example we expect it to be smooth if our target distribution is smooth. GANs, in contrast, are free to learn any arbitrary mapping between noise and images. This feature of diffusion models may make the learning

problem easier in some cases (since it is supervised), or harder in other cases (since there may be easier-to-learn maps which other methods could find).

3.7 *Remarks on Generalization*

In this tutorial, we have not discussed the learning-theoretic aspects of diffusion models: How do we learn properties of the underlying distribution, given only finite samples and bounded compute? These are fundamental aspects of learning, but are not yet fully understood for diffusion models; it is an active area of research[40].

To appreciate the subtlety here, suppose we learn a diffusion model using the classic strategy of Empirical Risk Minimization (ERM): we sample a finite train set from the underlying distribution, and optimize all regression functions w.r.t. this empirical distribution. The problem is, we should not *perfectly* minimize the empirical risk, because this would yield a diffusion model which only reproduces the train samples[41].

In general learning the diffusion model must be *regularized*, implicitly or explicitly, to prevent overfitting and memorization of the training data. When we train deep neural networks for use in diffusion models, this regularization often occurs implicitly: factors such as finite model size and optimization randomness prevent the trained model from perfectly memorizing its train set. We will revisit these factors (as sources of error) in Section 5.

This issue of memorizing training data has been seen "in the wild" in diffusion models trained on small image datasets, and it has been observed that memorization reduces as the training set size increases [Somepalli et al., 2023, Gu et al., 2023]. Additionally, memorization as been noted as a potential security and copyright issue for neural networks as in Carlini et al. [2023] where the authors found they can recover training data from stable diffusion with the right prompts.

Figure 6 demonstrates the effect of training set size, and shows the DDIM trajectories for a diffusion model trained using a 3 layer ReLU network. We see that the diffusion model on $N = 10$ samples "memorizes" its train set: its trajectories all collapse to one of the train points, instead of producing the underlying spiral distribution. As we add more samples, the model starts to generalize: the trajectories converge to the underlying spiral manifold. The trajectories also start to become more perpendicular the underlying manifold, suggesting that the low dimensional structure is being learned. We also note that in the $N = 10$ case where the diffusion model fails, it is not at all obvious a human would be able to identify the "correct" pattern from these samples, so generalization may be too much to expect.

[40] We recommend the introductions of Chen et al. [2022] and Chen et al. [2024b] for an overview of recent learning-theoretic results. This line of work includes e.g. De Bortoli et al. [2021], De Bortoli [2022], Lee et al. [2023], Chen et al. [2023, 2024a].

[41] This is not specific to diffusion models: any perfect generative model of the empirical distribution will always output a uniformly random train point, which is far-from-optimal w.r.t. the true underlying distribution.

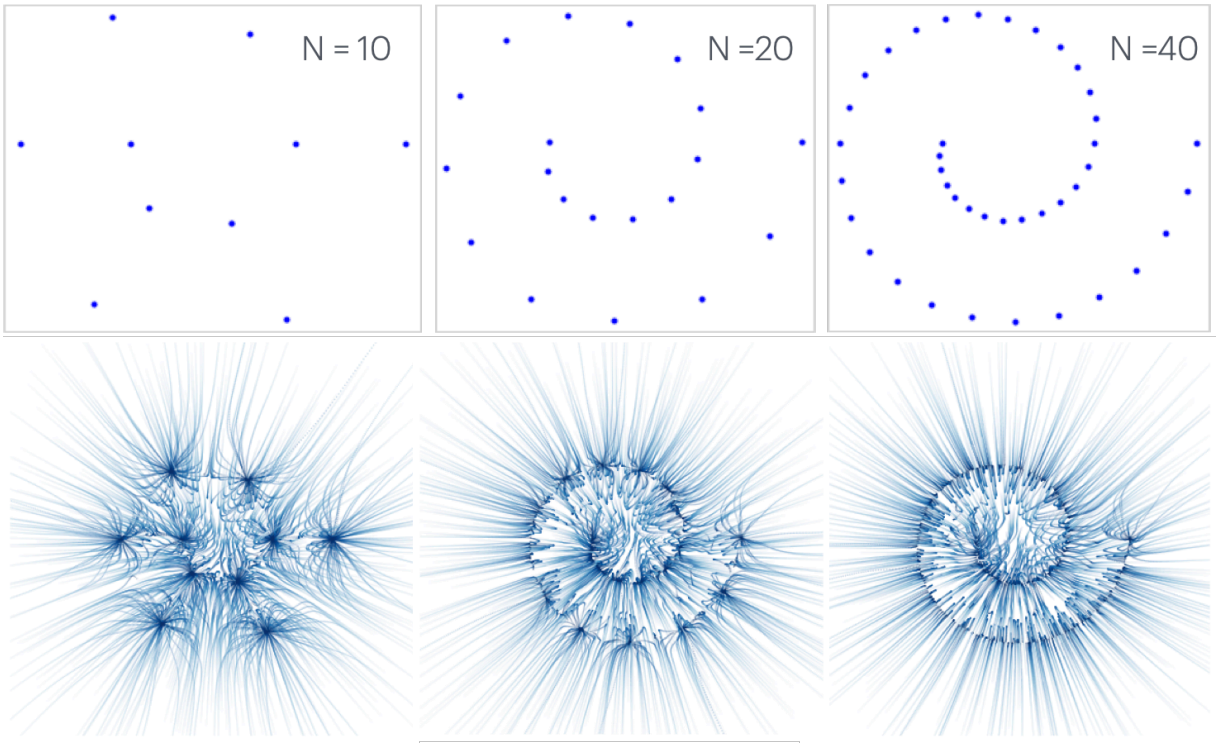

Figure 6: The DDIM trajectories (shaded by timestep t) for a spiral dataset. We compare the trajectories with $10, 20$, and 40 training samples. Note that as we add more training points (moving left to right) the diffusion algorithm begins to *learn* the underlying spiral and the trajectories look more perpendicular to the underlying manifold. The network used here is a 3 layer ReLU network with 128 neurons per layer.

4 Flow Matching

We now introduce the framework of flow matching [Peluchetti, 2022, Liu et al., 2022b,a, Lipman et al., 2023, Albergo et al., 2023]. Flow matching can be thought of as a generalization of DDIM, which allows for more flexibility in designing generative models— including for example the *rectified flows* (sometimes called *linear flows*) used by Stable Diffusion 3 [Liu et al., 2022a, Esser et al., 2024].

We have actually already seen the main ideas behind flow matching, in our analysis of DDIM in Section 3. At a high level, here is how we constructed a generative model in Section 3:

1. First, we defined how to generate a single point. Specifically, we constructed vector fields $\{v_t^{[a]}\}_t$ which, when applied for all time steps, transported a standard Gaussian distribution to an arbitrary delta distribution δ_a.
2. Second, we determined how to combine two vector fields into a single effective vector field. This lets us construct a transport from the standard Gaussian to *two* points (or, more generally, to a *distribution* over points — our target distribution).

Neither of these steps particularly require the Gaussian base distribution, or the Gaussian forward process (Equation 1). The second step of combining vector fields remains identical for any two arbitrary vector fields, for example.

So let's drop all the Gaussian assumptions. Instead, we will begin by thinking at a basic level about how to map between any two points x_0 and x_1. Then, we see what happens when the two points are sampled from arbitrary distributions p (data) and q (base), respectively. We will see that this point of view encompasses DDIM as a special case, but that it is significantly more general.

4.1 Flows

Let us first define the central notion of a *flow*. A *flow* is simply a collection of time-indexed vector fields $v = \{v_t\}_{t\in[0,1]}$. We should think of this as the velocity-field v_t of a gas at each time t, as we did earlier in Section 3.2. Any flow defines a trajectory taking initial points x_1 to final points x_0, by transporting the initial point along the velocity fields $\{v_t\}$.

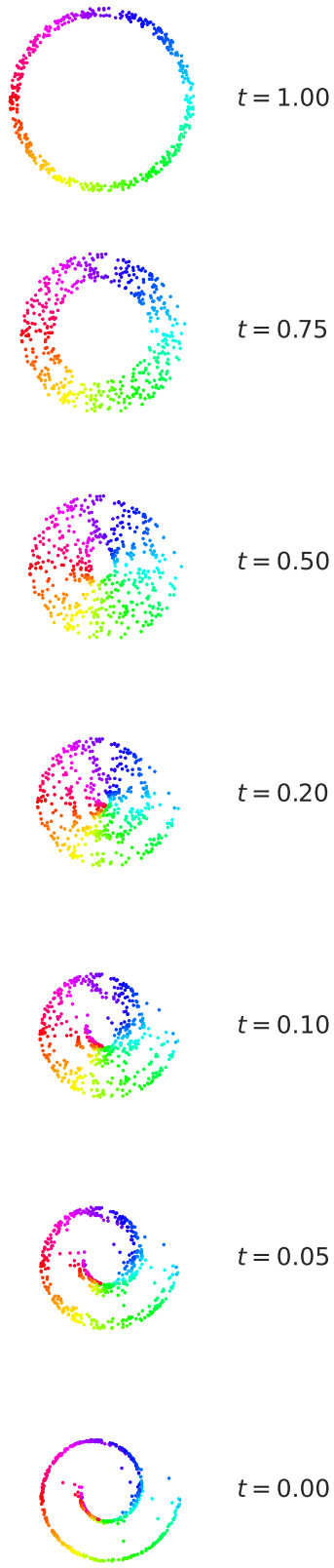

Figure 7: Running a flow which generates a spiral distribution (bottom) from an annular distribution (top).

Formally, for flow v and initial point x_1, consider the ODE[42]

$$\frac{dx_t}{dt} = -v_t(x_t), \tag{59}$$

with initial condition x_1 at time $t = 1$. We write

$$x_t := \text{RunFlow}(v, x_1, t) \tag{60}$$

to denote the solution to the flow ODE (Equation 59) at time t, terminating at final point x_0. That is, RunFlow is the result of transporting point x_1 along the flow v up to time t.

Just as flows define maps between initial and final points, they also define transports between entire *distributions*, by "pushing forward" points from the source distribution along their trajectories. If p_1 is a distribution on initial points[43], then applying the flow v yields the distribution on final points[44]

$$p_0 = \{\text{RunFlow}(v, x_1, t = 0)\}_{x_1 \sim p_1}. \tag{61}$$

We denote this process as $p_1 \overset{v}{\hookrightarrow} p_0$ meaning the flow v transports initial distribution p_1 to final distribution[45] p_0.

THE ULTIMATE GOAL OF FLOW MATCHING is to somehow learn a flow v^* which transports $q \overset{v^*}{\hookrightarrow} p$, where p is the target distribution and q is some easy-to-sample base distribution (such as a Gaussian). If we had this v^*, we could generate samples from our target p by first sampling $x_1 \sim q$, then running our flow with initial point x_1 and outputting the resulting final point x_0. The DDIM algorithm of Section 3 was actually a special case[46] of this, for a very particular choice of flow v^*. Now, how do we construct such flows in general?

4.2 *Pointwise Flows*

Our basic building-block will be a *pointwise flow* which just transports a single point x_1 to a point x_0. Intuitively, given an arbitrary path $\{x_t\}_{t\in[0,1]}$ that connects x_1 to x_0, a pointwise flow describes this trajectory by giving its velocity $v_t(x_t)$ at each point x_t along it (see Figure 8). Formally, a *pointwise flow between x_1 and x_0* is any flow $\{v_t\}_t$ that satisfies Equation 59 with boundary conditions x_1 and x_0 at times $t = 1, 0$ respectively. We denote such flows as $v^{[x_1,x_0]}$. Pointwise flows are not unique: there are many different choices of path between x_0 and x_1.

4.3 *Marginal Flows*

Suppose that for all pairs of points (x_1, x_0), we can construct an explicit pointwise flow $v^{[x_1,x_0]}$ that transports a source point x_1 to target

[42] The corresponding discrete-time analog is the iteration: $x_{t-\Delta t} \leftarrow x_t + v_t(x_t)\Delta t$, starting at $t = 1$ with initial point x_1.

[43] Notational warning: Most of the flow matching literature uses a reversed time convention, so $t = 1$ is the target distribution. We let $t = 0$ be the target distribution to be consistent with the DDPM convention.

[44] We could equivalently write this as the pushforward $\text{RunFlow}(v, \cdot, 0)\sharp p_1$.

[45] In our gas analogy, this means if we start with a gas of particles distributed according to p_1, and each particle follows the trajectory defined by v, then the final distribution of particles will be p_0.

[46] To connect to diffusion: The continuous-time limit of DDIM (58) is a flow with $v_t(x_t) = \frac{1}{2t}\,\mathbb{E}[x_0 - x_t|x_t]$. The base distribution p_1 is Gaussian. DDIM Sampling (algorithm 3) is a discretized method for evaluating RunFlow. DDPM Training (algorithm 2) is a method for learning $v^\star$ – but it relies on the Gaussian structure and differs somewhat from the flow-matching algorithm we will present in this chapter.

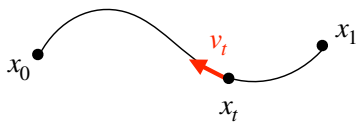

Figure 8: A pointwise flow $v_t^{[x_1,x_0]}$ transporting x_1 to x_0.

point x_0. For example, we could let x_t travel along a straight line from x_1 to x_0, or along any other explicit path. Recall in our gas analogy, this corresponds to an individual particle that moves between x_1 and x_0. Now, let us try to set up a collection of individual particles, such that at $t = 1$ the particles are distributed according to q, and at $t = 0$ they are distributed according to p. This is actually easy to do: We can pick any coupling[47] $\Pi_{q,p}$ between q and p, and consider particles corresponding to the pointwise flows $\{v^{[x_1,x_0]}\}_{(x_1,x_0)\sim\Pi_{q,p}}$. This gives us a distribution over pointwise flows (i.e. a collection of particle trajectories) with the desired behavior in aggregate.

We would like to combine all of these pointwise flows somehow, to get a single flow v^* that implements the same transport between distributions[48]. Our previous discussion[49] in Section 3 tells us how to do this: to determine the effective velocity $v_t^*(x_t)$, we should take a weighted-average of all individual particle velocities $v_t^{[x_1,x_0]}$, weighted by the probability that a particle at x_t was generated by the pointwise flow $v^{[x_1,x_0]}$. The final result is[50]

$$v_t^*(x_t) := \mathop{\mathbb{E}}_{x_0,x_1|x_t}[v_t^{[x_1,x_0]}(x_t) \mid x_t] \tag{64}$$

where the expectation is w.r.t. the joint distribution of (x_1, x_0, x_t) induced by sampling $(x_1, x_0) \sim \Pi_{q,p}$ and letting $x_t \leftarrow \text{RunFlow}(v^{[x_1,x_0]}, x_1, t)$.

At this point, we have a "solution" to our generative modeling problem in principle, but some important questions remain to make it useful in practice:

- Which pointwise flow $v^{[x_1,x_0]}$ and coupling $\Pi_{q,p}$ should we chose?
- How do we compute the marginal flow v^*? We cannot compute it from Equation (64) directly, because this would require sampling from $p(x_0 \mid x_t)$ for a given point x_t, which may be complicated in general.

We answer these in the next sections.

4.4 *A Simple Choice of Pointwise Flow*

We need an explicit choices of: pointwise flow, base distribution q, and coupling $\Pi_{q,p}$. There are many simple choices which would work[51].

The base distribution q can be essentially any easy-to-sample distribution. Gaussians are a popular choice but certainly not the only one— Figure 7 uses an annular base distribution, for example. As for the coupling $\Pi_{q,p}$ between the base and target distribution, the simplest choice is the independent coupling, i.e. sampling from p and q independently.

[47] A *coupling* $\Pi_{q,p}$ between q and p, specifies how to jointly sample pairs (x_1, x_0) of source and target points, such that x_0 is marginally distributed as p, and x_1 as q. The most basic coupling is the *independent coupling*, with corresponds to sampling x_1, x_0 independently.

[48] Why would we like this? As we will see later, it simplifies our learning problem: instead of having to learn the distribution of all the individual trajectories, we can instead just learn one velocity field representing their bulk evolution.

[49] Compare to Equation (49) in Section 3. A formal statement of how to combine flows is given in Appendix B.4.

[50] An alternate way of viewing this result at a high level is: we start with pointwise flows $v^{[x_1,x_0]}$ which transport delta distributions:

$$\delta_{x_1} \overset{v^{[x_1,x_0]}}{\hookrightarrow} \delta_{x_0}. \tag{62}$$

And then Equation (64) gives us a fancy way of "averaging these flows over x_1 and x_0", to get a flow v^* transporting

$$q = \mathop{\mathbb{E}}_{x_1\sim q}[\delta_{x_1}] \overset{v^*}{\hookrightarrow} \mathop{\mathbb{E}}_{x_0\sim p}[\delta_{x_0}] = p. \tag{63}$$

[51] Diffusion provides one possible construction, as we will see later in Section 4.6.

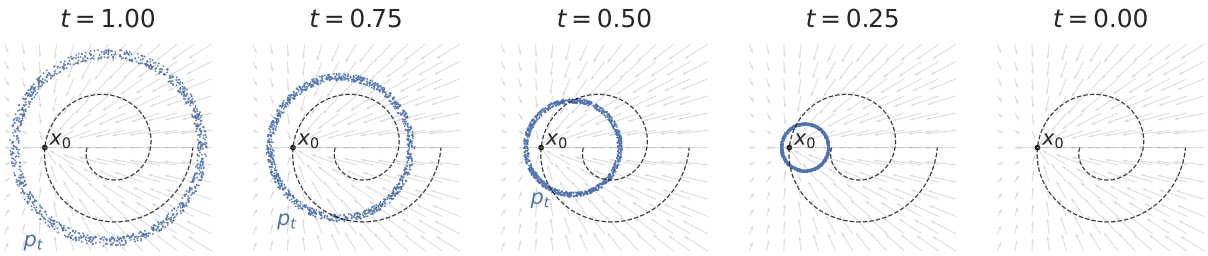

Figure 9: A marginal flow with linear pointwise flows, base distribution q uniform over an annulus, and target distribution p equal to a Dirac-delta at x_0. (This can also be thought of as the average over x_1 of the pointwise linear flows from $x_1 \sim q$ to a fixed x_0). Gray arrows depict the flow field at different times t. The leftmost $(t = 1)$ plot shows samples from the base distribution q. Subsequent plots show these samples transported by the flow at intermediate times t, The final $(t = 0)$ plot shows all points collapsed to the target x_0. This particular x_0 happens to be one point on the spiral distribution of Figure 7.

For a pointwise flow, arguably the simplest construction is a *linear pointwise flow*:

$$v_t^{[x_1,x_0]}(x_t) = x_0 - x_1, \tag{65}$$

$$\implies \text{RunFlow}(v^{[x_1,x_0]}, x_1, t) = tx_1 + (1-t)x_0 \tag{66}$$

which simply linearly interpolates between x_1 and x_0 (and corresponds to the choice made in Liu et al. [2022a]). In Figure 9 we visualize a marginal flow composed of linear pointwise flows, the same annular base distribution q of Figure 7, and target distribution equal to a point-mass $(p = \delta_{x_0})$[52].

[52] A marginal distribution with a point-mass target distribution – or equivalently the average of pointwise flows over the the base distribution only – is sometimes called a *(one-sided) conditional flow* [Lipman et al., 2023].

4.5 *Flow Matching*

Now, the only remaining problem is that naively evaluating v^* using Equation (64) requires sampling from $p(x_0 \mid x_t)$ for a given x_t. If we knew how do this for $t = 1$, we would have already solved the generative modeling problem!

Fortunately, we can take advantage of the same trick from DDPM: it is enough for us to be able to sample from the joint distribution (x_0, x_t), and then solve a regression problem. Similar to DDPM, the conditional expectation function in Equation (64) can be written as a regressor[53]:

[53] This result is analogous to Theorem 2 in Lipman et al. [2023], but ours is for a two-sided flow.

$$v_t^*(x_t) := \mathop{\mathbb{E}}_{x_0,x_1|x_t}[v_t^{[x_1,x_0]}(x_t) \mid x_t] \tag{67}$$

$$\implies v_t^* = \mathop{\text{argmin}}_{f:\mathbb{R}^d\to\mathbb{R}^d} \mathop{\mathbb{E}}_{(x_0,x_1,x_t)} ||f(x_t) - v_t^{[x_1,x_0]}(x_t)||_2^2, \tag{68}$$

(by using the generic fact that $\text{argmin}_f \mathbb{E}\,||f(x) - y||^2 = \mathbb{E}[y \mid x]$).

In words, Equation (68) says that to compute the loss of a model f_θ for a fixed time t, we should:

1. Sample source and target points (x_1, x_0) from their joint distribution.
2. Compute the point x_t deterministically, by running[54] the pointwise flow $v^{[x_1,x_0]}$ starting from point x_1 up to time t.
3. Evaluate the model's prediction at x_t, as $f_\theta(x_t)$. Evaluate the deterministic vector $v_t^{[x_1,x_0]}(x_t)$. Then compute L2 loss between these two quantities.

[54] If we chose linear pointwise flows, for example, this would mean $x_t \leftarrow tx_1 + (1-t)x_0$, via Equation (66).

To sample from the trained model (our estimate of v_t^*), we first sample a source point $x_1 \sim q$, then transport it along the learnt flow to a target sample x_0. Pseudocode listings 4 and 5 give the explicit procedures for training and sampling from flow-based models (including the special case of linear flows for concreteness; matching Algorithm 1 in Liu et al. [2022a].).

Summary

To summarize, here is how to learn a flow-matching generative model for target distribution p.

The Ingredients. We first choose:

1. A source distribution q, from which we can efficiently sample (e.g. a standard Gaussian).
2. A *coupling* $\Pi_{q,p}$ between q and p, which specifies a way to jointly sample a pair of source and target points (x_1, x_0) with marginals q and p respectively. A standard choice is the independent coupling, i.e. sample $x_1 \sim q$ and $x_0 \sim p$ independently.
3. For all pairs of points (x_1, x_0), an explicit *pointwise flow* $v^{[x_1,x_0]}$ which transports x_1 to x_0. We must be able to efficiently compute the vector field $v_t^{[x_1,x_0]}$ at all points.

These ingredients determine, in theory, a marginal vector field v^* which transports q to p:

$$v_t^*(x_t) := \mathop{\mathbb{E}}_{x_0,x_1|x_t}[v_t^{[x_1,x_0]}(x_t) \mid x_t] \tag{69}$$

where the expectation is w.r.t. the joint distribution:

$$\begin{aligned}(x_1, x_0) &\sim \Pi_{q,p} \\ x_t &:= \text{RunFlow}(v^{[x_1,x_0]}, x_1, t).\end{aligned}$$

Training. Train a neural network f_θ by backpropogating the stochastic loss function computed by Pseudocode 4. The optimal function for this expected loss is: $f_\theta(x_t, t) = v_t^*(x_t)$.

Sampling. Run Pseudocode 5 to generate a sample x_0 from (approimately) the target distribution p.

Pseudocode 4: Flow-matching train loss, generic pointwise flow [or linear flow]

Input: Neural network f_θ
Data: Sample-access to coupling $\Pi_{q,p}$;
Pointwise flows $\{v_t^{[x_1,x_0]}\}$ for all x_1, x_0.
Output: Stochastic loss L

$(x_1, x_0) \leftarrow \text{Sample}(\Pi_{q,p})$
$t \leftarrow \text{Unif}[0,1]$
$x_t \leftarrow \underbrace{\text{RunFlow}(v^{[x_1,x_0]}, x_1, t)}_{tx_1+(1-t)x_0}$
$L \leftarrow \left\| f_\theta(x_t, t) - \underbrace{v_t^{[x_1,x_0]}(x_t)}_{(x_0-x_1)} \right\|_2^2$
return L

Pseudocode 5: Flow-matching sampling

Input: Trained network f_θ
Data: Sample-access to base distribution q;
step-size Δt.
Output: Sample from target distribution p.

$x_1 \leftarrow \text{Sample}(q)$
for $t = 1,\ (1-\Delta t), (1-2\Delta t), \ldots, \Delta t$ **do**
  $x_{t-\Delta t} \leftarrow x_t + f_\theta(x_t, t)\Delta t$
end
return x_0

4.6 *DDIM as Flow Matching [Optional]*

The DDIM algorithm of Section 3 can be seen as a special case of flow matching, for a particular choice of pointwise flows and coupling. We describe the exact correspondence here, which will allow us to notice an interesting relation between DDIM and linear flows.

We claim DDIM is equivalent to flow-matching with the following parameters:

1. **Pointwise Flows:** Either of the two equivalent pointwise flows:

$$v_t^{[x_1,x_0]}(x_t) := \frac{1}{2t}(x_t - x_0) \tag{70}$$

or

$$v_t^{[x_1,x_0]}(x_t) := \frac{1}{2\sqrt{t}}(x_0 - x_1), \tag{71}$$

which both generate the trajectory[55]:

$$x_t = x_0 + (x_1 - x_0)\sqrt{t}. \tag{72}$$

2. **Coupling:** The "diffusion coupling" – that is, the joint distribution on (x_0, x_1) generated by

$$x_0 \sim p; \quad x_1 \leftarrow x_0 + \mathcal{N}(0, \sigma_q^2). \tag{73}$$

[55] See Appendix B.6 for details on why (70) and (71) are equivalent along their trajectories.

This claim is straightforward to prove (see Appendix B.5), but the implication is somewhat surprising: we can recover the DDIM trajectories (which are not straight in general) as a combination of the *straight* pointwise trajectories in Equation (72). In fact, the DDIM trajectories are exactly equivalent to flow-matching trajectories for the above linear flows, with a different scaling of time ($\sqrt{t}$ vs. t)[56].

Claim 4 (DDIM as Linear Flow; Informal)**.** *The DDIM sampler (Algorithm 2) is equivalent, up to time-reparameterization, to the marginal flow produced by linear pointwise flows (Equation 65) with the diffusion coupling (Equation 73).*

A formal statement of this claim[57] is provided in Appendix B.7.

[56] DDIM at time t corresponds to the linear flow at time $\sqrt{t}$; thus linear flows are "slower" than DDIM when t is small. This may be beneficial for linear flows in practice (speculatively).

[57] In practice, linear flows are most often instantiated with the independent coupling, not the above "diffusion coupling." However, for large enough terminal variance σ_q^2, the diffusion coupling is close to independent. Therefore, Claim 4 tells us that the common practice in flow matching (linear flows with a Gaussian terminal distribution and independent coupling) is nearly equivalent to standard DDIM, with a different time schedule. Finally, for the experts: this is a claim about the "variance exploding" version of DDIM, which is what we use throughout. Claim 4 is false for variance-preserving DDIM.

4.7 *Additional Remarks and References [Optional]*

- See Figure 11 for a diagram of the different methods described in this tutorial, and their relations.
- We highly recommend the flow-matching tutorial of Fjelde et al. [2024], which includes helpful visualizations of flows, and uses notation more consistent with the current literature.
- As a curiosity, note that we never had to define an explicit "forward process" for flow-matching, as we did for Gaussian diffusion. Rather, it was enough to define the appropriate "reverse processes" (via flows).
- What we called *pointwise flows* are also called *two-sided conditional flows* in the literature, and was developed in Albergo and Vanden-Eijnden [2022], Pooladian et al. [2023], Liu et al. [2022a], Tong et al. [2023].
- Albergo et al. [2023] define the framework of *stochastic interpolants*, which can be thought of as considering stochastic pointwise flows, instead of only deterministic ones. Their framework strictly generalizes both DDPM and DDIM.
- See Stark et al. [2024] for an interesting example of non-standard flows. They derive a generative model for discrete spaces by embedding into a continuous space (the probability simplex), then constructing a special flow on these simplices.

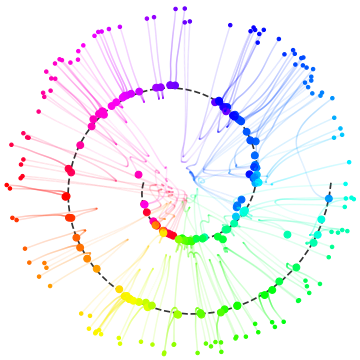

Figure 10: The trajectories of individual samples $x_1 \sim q$ for the flow in Figure 7.

5 Diffusion in Practice

To conclude, we mention some aspects of diffusion which are important in practice, but were not covered in this tutorial.

Samplers in Practice. Our DDPM and DDIM samplers (algorithms 2 and 3) correspond to the samplers presented in Ho et al. [2020] and Song et al. [2021], respectively, but with different choice of schedule and parametrization (see footnote 13). DDPM and DDIM were some of the earliest samplers to be used in practice, but since then there has been significant progress in samplers for fewer-step generation (which is crucial since each step requires a typically-expensive model forward-pass).[58] In sections 2.4 and 3.5, we showed that DDPM and DDIM can be seen as discretizations of the reverse SDE and Probability Flow ODE, respectively. The SDE and ODE perspectives automatically lead to many samplers corresponding to different black-box SDE and ODE numerical solvers (such as Euler, Heun, and Runge-Kutta). It is also possible to take advantage of the specific structure of the diffusion ODE, to improve upon black-box solvers [Lu et al., 2022a,b, Zhang and Chen, 2023].

Noise Schedules. The *noise schedule* typically refers to σ_t, which determines the amount of noise added at time t of the diffusion process. The simple diffusion (1) has $p(x_t) \sim \mathcal{N}(x_0, \sigma_t^2)$ with $\sigma_t \propto \sqrt{t}$. Notice that the variance of x_t increases at every timestep.[59]

In practice, schedules with controlled variance are often preferred. One of the most popular schedules, introduced in Ho et al. [2020], uses a time-dependent variance and scaling such that the variance of x_t remains bounded. Their discrete update is

$$x_t = \sqrt{1-\beta(t)}x_{t-1} + \sqrt{\beta(t)}\varepsilon_t; \quad \varepsilon_t \sim \mathcal{N}(0,1), \tag{74}$$

where $0 < \beta(t) < 1$ is chosen so that x_t is (very close to) clean data at $t = 1$ and pure noise at $t = T$.

The general SDE (26) introduced in 2.4 offers additional flexibility. Our simple diffusion (1) has $f = 0, g = \sigma_q$, while the diffusion (74) of Ho et al. [2020] has $f = -\frac{1}{2}\beta(t), g = \sqrt{\beta(t)}$. Karras et al. [2022] reparametrize the SDE in terms of an overall scaling $s(t)$ and variance $\sigma(t)$ of x_t, as a more interpretable way to think about diffusion designs, and suggest a schedule with $s(t) = 1, \sigma(t) = t$ (which corresponds to $f = 0, g = \sqrt{2t}$). Generally, the choice of f, g, or equivalently s, σ, offers a convenient way to explore the design-space of possible schedules.

[58] Even the best samplers still require around 10 sampling steps, which may be impractical. A variety of *time distillation* methods seek to train one-step-generator student models to match the output of diffusion teacher models, with the goal of high-quality sampling in one (or few) steps. Some examples include consistency models [Song et al., 2023b] and adversarial distillation methods [Lin et al., 2024, Xu et al., 2023, Sauer et al., 2024]. Note, however, that the distilled models are no longer diffusion models, nor are their samplers (even if multi-step) diffusion samplers.

[59] Song et al. [2020] made the distinction between "variance-exploding" (VE) and "variance-preserving" (VP) schedules while comparing SMLD [Song and Ermon, 2019] and DDPM [Ho et al., 2020]. The terms VE and VP often refer specifically to SMLD and DDPM, respectively. Our diffusion (1) could also be called a variance-exploding schedule, though our noise schedule differs from the one originally proposed in Song and Ermon [2019].

Likelihood Interpretations and VAEs. One popular and useful interpretation of diffusion models is the Variational Auto Encoder (VAE) perspective[60]. Briefly, diffusion models can be viewed as a special case of a deep hierarchical VAE, where each diffusion timestep corresponds to one "layer" of the VAE decoder. The corresponding VAE encoder is given by the forward diffusion process, which produces the sequence of noisy $\{x_t\}$ as the "latents" for input x. Notably, the VAE encoder here is not learnt, unlike usual VAEs. Because of the Markovian structure of the latents, each layer of the VAE decoder can be trained in isolation, without forward/backward passing through all previous layers; this helps with the notorious training instability of deep VAEs. We recommend the tutorials of Turner [2021] and Luo [2022] for more details on the VAE perspective.

[60] This was actually the original approach to derive the diffusion objective function, in Sohl-Dickstein et al. [2015] and also Ho et al. [2020].

One advantage of the VAE interpretation is, it gives us an estimate of the *data likelihood* under our generative model, by using the standard Evidence-Based-Lower-Bound (ELBO) for VAEs. This allows us to train diffusion models directly using a maximum-likelihood objective. It turns out that the ELBO for the diffusion VAE reduces to exactly the L2 regression loss that we presented, but with a particular *time-weighting* that weights the regression loss differently at different time-steps t. For example, regression errors at large times t (i.e. at high noise levels) may need to be weighted differently from errors at small times, in order for the overall loss to properly reflect a likelihood.[61] The best choice of time-weighting in practice, however, is still up for debate: the "principled" choice informed by the VAE interpretation does not always produce the best generated samples[62]. See Kingma and Gao [2023] for a good discussion of different weightings and their effect.

[61] See also Equation (5) in Kadkhodaie et al. [2024] for a simple bound on KL divergence between the true distribution and generated distribution, in terms of regression excess risks.

[62] For example, Ho et al. [2020] drops the time-weighting terms, and just uniformly weights all timesteps.

Parametrization: x_0 / ε / v -prediction. Another important practical choice is which of several closely-related quantities – partially-denoised data, fully-denoised data, or the noise itself – we ask the network to predict.[63] Recall that in DDPM Training (Algorithm 1), we asked the network f_θ to learn to predict $\mathbb{E}[x_{t-\Delta t}|x_t]$ by minimizing $\|f_\theta(x_t,t) - x_{t-\Delta t}\|_2^2$. However, other parametrizations are possible. For example, recalling that $\mathbb{E}[x_{t-\Delta t} - x_t|x_t] \overset{\text{eq. 23}}{=} \frac{\Delta t}{t}\mathbb{E}[x_0 - x_t|x_t]$, we see that that

[63] More accurately, the network always predicts conditional expectations of these quantities.

$$\min_\theta \quad \|f_\theta(x_t,t) - x_0\|_2^2 \implies f_\theta^\star(x_t,t) = \mathbb{E}[x_0|x_t]$$

is a (nearly) equivalent problem, which is often called x_0-*prediction*.[64] The objectives differ only by a time-weighting factor of $\frac{1}{t}$. Similarly, defining the *noise* $\varepsilon_t = \frac{1}{\sigma_t}\mathbb{E}[x_0 - x_t|x_t]$, we see that we could alternatively ask the the network to predict $\mathbb{E}[\varepsilon_t|x_t]$: this is usually called

[64] This corresponds to the variance-reduced algorithm (6).

ε-prediction. Another parametrization, *v-prediction*, asks the model to predict $v = \alpha_t \varepsilon - \sigma_t x_0$ [Salimans and Ho, 2022] – mostly predicting data for high noise-levels and mostly noise for low noise-levels. All the parametrizations differ only by time-weightings (see Appendix B.10 for more details).

Although the different time-weightings do not affect the optimal solution, they do impact training as discussed above. Furthermore, even if the time-weightings are adjusted to yield equivalent problems in principle, the different parametrizations may behave differently in practice, since learning is not perfect and certain objectives may be more robust to error. For example, x_0-prediction combined with a schedule that places a lot of weight on low noise levels may not work well in practice, since for low noise the identity function can achieve a relatively low objective value, but clearly is not what we want.

Sources of Error. Finally, when using diffusion and flow models in practice, there are a number of sources of error which prevent the learnt generative model from exactly producing the target distribution. These can be roughly segregated into training-time and sampling-time errors.

1. Train-time error: Regression errors in learning the population-optimal regression function. The regression objective is the marginal flow v_t^* in flow-matching, or the scores $\mathbb{E}[x_0 \mid x_t]$ in diffusion models. For each fixed time t, this a standard kind of statistical error. It depends on the neural network architecture and size as well as the number of samples, and can be decomposed further into approximation and estimation errors in the usual way (e.g. see Advani et al. [2020, Sec. 4] decomposing a 2-layer network into approximation error and over-fitting error).

2. Sampling-time error: Discretization errors from using finite step-sizes Δt. This error is exactly the discretization error of the ODE or SDE solver used in sampling. These errors manifest in different ways: for DDPM, this reflects the error in using a Gaussian approximation of the reverse process (i.e. Fact 1 breaks for large σ). For DDIM and flow matching, it reflects the error in simulating continuous-time flows in discrete time.

These errors interact and compound in nontrivial ways, which are not yet fully understood. For example, it is not clear exactly how train-time error in the regression estimates translates into distributional error of the entire generative model. (And this question itself is complicated, since it is not always clear what type of distributional divergence we care about in practice). Interestingly, these "errors" can also have a beneficial effect on small train sets, because they act

as a kind of *regularization* which prevents the diffusion model from just memorizing the train samples (as discussed in Section 3.7).

Conclusion

We have now covered the basics of diffusion models and flow matching. This is an active area of research, and there are many interesting aspects and open questions which we did not cover (see Page 36 for recommended reading). We hope the foundations here equip the reader to understand more advanced topics in diffusion modeling, and perhaps contribute to the research themselves.

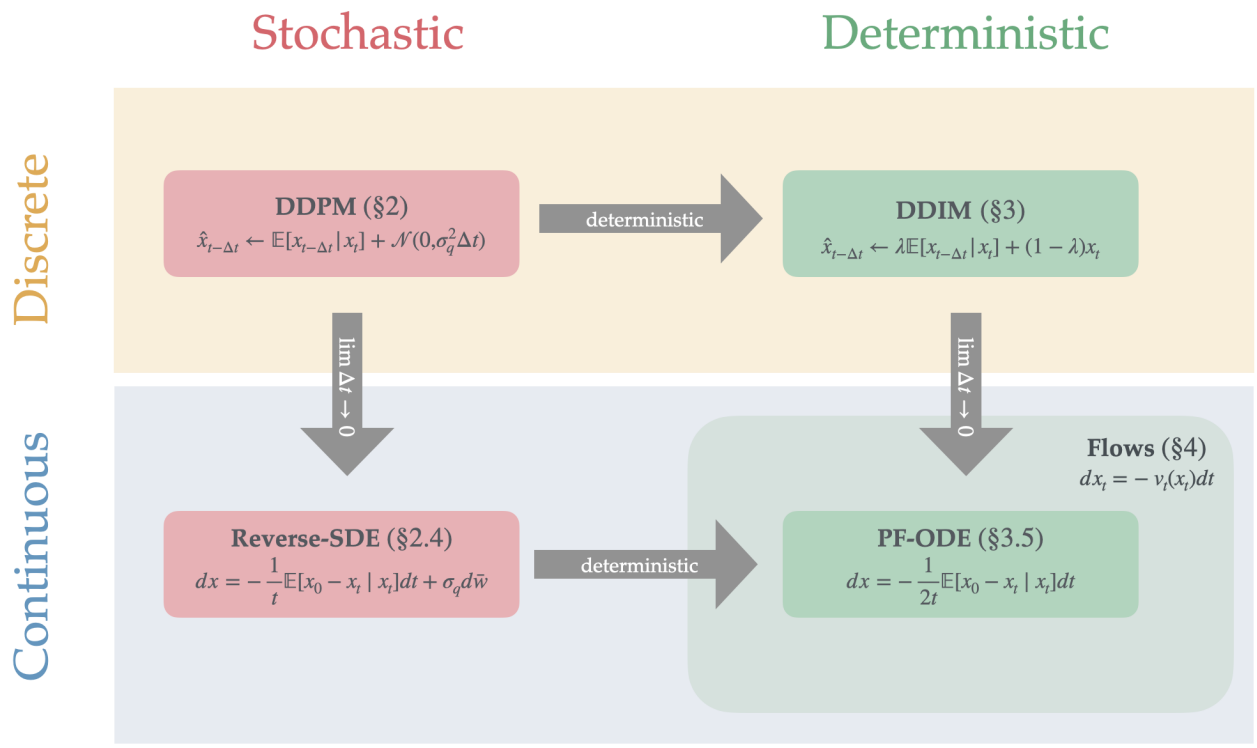

Figure 11: Commutative diagram of the different reverse samplers described in this tutorial, and their relations. Each deterministic sampler produces identical marginal distributions as its stochastic counterpart. There are also various ways to construct stochastic versions of flows, which are not pictured here (e.g. Albergo et al. [2023]).

A Additional Resources

Several other helpful resources for learning diffusion (tutorials, blogs, papers), roughly in order of mathematical background required.

1. **Perspectives on diffusion.**
   Dieleman [2023]. (Webpage.)

   Overview of many interpretations of diffusion, and techniques.

2. **Tutorial on Diffusion Models for Imaging and Vision.**
   Chan [2024]. (49 pgs.)

   More focus on intuitions and applications.

3. **Interpreting and improving diffusion models using the euclidean distance function.**
   Permenter and Yuan [2023]. (Webpage.)

   Distance-field interpretation. See accompanying blog with simple code [Yuan, 2024].

4. **On the Mathematics of Diffusion Models.**
   McAllester [2023]. (4 pgs.)

   Short and accessible.

5. **Building Diffusion Model's theory from ground up**
   Das [2024]. (Webpage.)

   ICLR 2024 Blogposts Track. Focus on SDE and score-matching perspective.

6. **Denoising Diffusion Models: A Generative Learning Big Bang.**
   Song, Meng, and Vahdat [2023a]. (Video, 3 hrs.)

   CVPR 2023 tutorial, with recording.

7. **Diffusion Models From Scratch.**
   Duan [2023]. (Webpage, 10 parts.)

   Fairly complete on topics, includes: DDPM, DDIM, Karras et al. [2022], SDE/ODE solvers. Includes practical remarks and code.

8. **Understanding Diffusion Models: A Unified Perspective.**
   Luo [2022]. (22 pgs.)

   Focus on VAE interpretation, with explicit math details.

9. **Demystifying Variational Diffusion Models.**
   Ribeiro and Glocker [2024]. (44 pgs.)

   Focus on VAE interpretation, with explicit math details.

10. **Diffusion and Score-Based Generative Models.**
    Song [2023]. (Video, 1.5 hrs.)

    Discusses several interpretations, applications, and comparisons to other generative modeling methods.

11. **Deep Unsupervised Learning using Nonequilibrium Thermodynamics**
    Sohl-Dickstein, Weiss, Maheswaranathan, and Ganguli [2015]. (9 pgs + Appendix)

Original paper introducing diffusion models for ML. Includes unified description of discrete diffusion (i.e. diffusion on discrete state spaces).

12. **An Introduction to Flow Matching.**
Fjelde, Mathieu, and Dutordoir [2024]. (Webpage.)

Insightful figures and animations, with rigorous mathematical exposition.

13. **Elucidating the Design Space of Diffusion-Based Generative Models.**
Karras, Aittala, Aila, and Laine [2022]. (10 pgs + Appendix.)

Discusses the effect of various design choices such as noise schhedule, parameterization, ODE solver, etc. Presents a generalized framework that captures many choices.

14. **Denoising Diffusion Models**
Peyré [2023]. (4 pgs.)

Fast-track through the mathematics, for readers already comfortable with Langevin dynamics and SDEs.

15. **Generative Modeling by Estimating Gradients of the Data Distribution.**
Song, Sohl-Dickstein, Kingma, Kumar, Ermon, and Poole [2020]. (9 pgs + Appendix.)

Presents the connections between SDEs, ODEs, DDIM, and DDPM.

16. **Stochastic Interpolants: A Unifying Framework for Flows and Diffusions.**
Albergo, Boffi, and Vanden-Eijnden [2023]. (46 pgs + Appendix.)

Presents a general framework that captures many diffusion variants, and learning objectives. For readers comfortable with SDEs

17. **Sampling, Diffusions, and Stochastic Localization.**
Montanari [2023]. (22 pgs + Appendix.)

Presents diffusion as a special case of "stochastic localization," a technique used in high-dimensional statistics to establish mixing of Markov chains.

B Omitted Derivations

B.1 KL Error in Gaussian Approximation of Reverse Process

Here we prove Lemma 1, restated below.

Lemma 2. *Let $p(x)$ be an arbitrary density over $\mathbb{R}$, with bounded 1st to 4th order derivatives. Consider the joint distribution (x_0, x_1), where $x_0 \sim p$ and $x_1 \sim x_0 + \mathcal{N}(0, \sigma^2)$. Then, for any conditioning $z \in \mathbb{R}$ we have*

$$\mathrm{KL}\left(\mathcal{N}(\mu_z, \sigma^2) \;||\; p_{x_0|x_1}(\cdot \mid x_1 = z)\right) \leq O(\sigma^4) \tag{75}$$

where

$$\mu_z := z + \sigma^2 \nabla \log p(z). \tag{76}$$

Proof. WLOG, we can take $z = 0$. We want to estimate the KL:

$$KL(\mathcal{N}(\mu, \sigma^2)||p(x_0 = \cdot \mid x_1 = 0)) \tag{77}$$

where we will let μ be arbitrary for now.

Let $q := \mathcal{N}(\mu, \sigma^2)$, and $p(x) =: \exp(F(x))$. We have $x_1 \sim p \star \mathcal{N}(0, \sigma^2)$. This implies:

$$p(x_1 = x) = \mathop{\mathbb{E}}_{\eta \sim \mathcal{N}(0,\sigma^2)}[p(x + \eta)]. \tag{78}$$

Let us first expand the logs of the two distributions we are comparing:

$$\begin{aligned}
&\log p(x_0 = x \mid x_1 = 0) && (79)\\
&= \log p(x_1 = 0 | x_0 = x) + \log p(x_0 = x) - \log p(x_1 = 0) && (80)\\
&= -\log(\sigma\sqrt{2\pi}) - 0.5x^2\sigma^{-2} + \log p(x_0 = x) - \log p(x_1 = 0) && (81)\\
&= -\log(\sigma\sqrt{2\pi}) - 0.5x^2\sigma^{-2} + F(x) - \log p(x_1 = 0) && (82)\\
& && (83)
\end{aligned}$$

And also:

$$\log q(x) = -\log(\sigma\sqrt{2\pi}) - 0.5(x - \mu)^2\sigma^{-2} \tag{84}$$

Now we can expand the KL:

$$
\begin{aligned}
&KL(q||p(x_0 = \cdot \mid x_1 = 0)) && (85)\\
&= \mathop{\mathbb{E}}_{x\sim q}[\log q(x) - \log p(x_0 = x \mid x_1 = 0)] && (86)\\
&= \mathop{\mathbb{E}}_{x\sim q}[-\log(\sigma\sqrt{2\pi}) - 0.5(x-\mu)^2\sigma^{-2} - (-\log(\sigma\sqrt{2\pi}) - 0.5x^2\sigma^{-2} + F(x) - \log p(x_1 = 0))] && (87)\\
&= \mathop{\mathbb{E}}_{x\sim q}[-0.5(x-\mu)^2\sigma^{-2} + 0.5x^2\sigma^{-2} - F(x) + \log p(x_1 = 0))] && (88)\\
&= \mathop{\mathbb{E}}_{\eta\sim\mathcal{N}(0,\sigma^2);x=\mu+\eta}[-0.5\eta^2\sigma^{-2} + 0.5x^2\sigma^{-2} - F(x) + \log p(x_1 = 0))] && \text{(work)}\\
&= -0.5\,\mathbb{E}[\eta^2]\sigma^{-2} + 0.5\,\mathbb{E}[x^2]\sigma^{-2} - \mathbb{E}[F(x)] + \log p(x_1 = 0))] && (89)\\
&= -0.5\sigma^2\sigma^{-2} + 0.5(\sigma^2 + \mu^2)\sigma^{-2} - \mathbb{E}[F(x)] + \log p(x_1 = 0))] && (90)\\
&= 0.5\mu^2\sigma^{-2} + \log p(x_1 = 0) - \mathop{\mathbb{E}}_{x\sim q}[F(x)] && (91)\\
&\approx 0.5\mu^2\sigma^{-2} + \log p(x_1 = 0) - \mathop{\mathbb{E}}_{x\sim q}[F(0) + F'(0)x + 0.5F''(0)x^2 + O(x^3) + O(x^4)] && (92)\\
&= \log p(x_1 = 0) + 0.5\mu^2\sigma^{-2} - F(0) - F'(0)\mu - 0.5F''(0)(\mu^2 + \sigma^2) + O(\sigma^2\mu + \mu^2 + \sigma^4) && (93)
\end{aligned}
$$

We will now estimate the first term, $\log p(x_1 = 0)$:

$$
\begin{aligned}
&\log p(x_1 = 0) && (94)\\
&= \log \mathop{\mathbb{E}}_{\eta\sim\mathcal{N}(0,\sigma^2)}[p(\eta)] && (95)\\
&= \log \mathop{\mathbb{E}}_{\eta\sim\mathcal{N}(0,\sigma^2)}[p(0) + p'(0)\eta + 0.5p''(0)\eta^2 + O(\eta^3) + O(\eta^4)] && (96)\\
&= \log\left(p(0) + 0.5p''(0)\sigma^2 + O(\sigma^4)\right) && (97)\\
&= \log p(0) + \frac{0.5p''(0)\sigma^2 + O(\sigma^4)}{p(0)} + O(\sigma^4) && \text{(Taylor expand } \log(p(0)+\varepsilon) \text{ around } p(0))\\
&= \log p(0) + 0.5\sigma^2\frac{p''(0)}{p(0)} + O(\sigma^4) && (98)
\end{aligned}
$$

To compute the derivatives of p, observe that:

$$
\begin{aligned}
F(x) &= \log p(x) && (99)\\
\implies F'(x) &= p'(x)/p(x) && (100)\\
\implies F''(x) &= p''(x)/p(x) - (p'(x)/p(x))^2 && (101)\\
&= p''(x)/p(x) - (F'(x))^2 && (102)\\
\implies p''(x)/p(x) &= F''(x) + (F'(x))^2 && (103)
\end{aligned}
$$

Thus, continuing from line (98):

$$
\begin{aligned}
\log p(x_1 = 0) &= \log p(0) + 0.5\sigma^2\frac{p''(0)}{p(0)} + O(\sigma^4) && (104)\\
&= F(0) + 0.5\sigma^2(F''(0) - F'(0)^2) + O(\sigma^4) && \text{(by Line 103)}
\end{aligned}
$$

We can now plug this estimate of $\log p(x_1 = 0)$ into Line (93). We omit the argument (0) from F for simplicity:

$$KL(q||p(x_0 = \cdot \mid x_1 = 0)) \tag{105}$$
$$= \boxed{\log p(x_1 = 0)} + 0.5\mu^2\sigma^{-2} - F - F'\mu - 0.5F''(\mu^2 + \sigma^2) + O(\mu^4 + \sigma^4) \tag{106}$$
$$= F + 0.5\sigma^2(F'' + F'^2) + 0.5\mu^2\sigma^{-2} - F - F'\mu - 0.5F''(\mu^2 + \sigma^2) + O(\mu^4 + \sigma^4) \tag{107}$$
$$= +0.5\sigma^2F'' + 0.5\sigma^2F'^2 + 0.5\mu^2\sigma^{-2} - F'\mu - 0.5F''\mu^2 - 0.5F''\sigma^2 + O(\mu^4 + \sigma^4) \tag{108}$$
$$= -F'\mu + 0.5\mu^2\sigma^{-2} + 0.5F'^2\sigma^2 - 0.5F''\mu^2 + O(\mu^4 + \sigma^4) \tag{109}$$

Up to this point, μ was arbitrary. We now set

$$\mu_* := F'(0)\sigma^2. \tag{110}$$

And continue:

$$KL(q||p(x_0 = \cdot \mid x_1 = 0)) \tag{111}$$
$$= -F'\mu_* + 0.5\mu_*^2\sigma^{-2} + 0.5F'^2\sigma^2 - 0.5F''\mu_*^2 + O(\mu_*^4 + \sigma^4) \tag{112}$$
$$= -F'^2\sigma^2 + 0.5F'^2\sigma^2 + 0.5F'^2\sigma^2 + O(\sigma^4) \tag{113}$$
$$= O(\sigma^4) \tag{114}$$

as desired.

□

Notice that our choice of μ_* in the above proof was crucial; for example if we had set $\mu_* = 0$, the $\Omega(\sigma^2)$ terms in Line (113) would not have cancelled out.

B.2 SDE proof sketches

Here is sketch of the proof of the equivalence of the SDE and Probability Flow ODE, which relies on the equivalence of the SDE to a Fokker-Planck equation. (See Song et al. [2020] for full proof.)

Proof.

$$\begin{aligned}
& dx = f(x,t)dt + g(t)dw \\
\iff & \frac{\partial p_t(x)}{\partial t} = -\nabla_x(f p_t) + \frac{1}{2}g^2\nabla_x^2 p_t \quad \text{(FP)} \\
& = -\nabla_x(f p_t) + \frac{1}{2}g^2\nabla_x(p_t \nabla_x \log p_t) \\
& = -\nabla_x\{(f - \frac{1}{2}g^2\nabla_x \log p_t)p_t\} \\
& = -\nabla_x\{\tilde{f}(x,t)p_t(x)\}, \quad \tilde{f}(x,t) = f(x,t) - \frac{1}{2}g(t)^2\nabla_x \log p_t(x) \\
\implies & dx = \tilde{f}(x,t)dt
\end{aligned}$$

□

The equivalence of the SDE and Fokker-Planck equations follows from Itô's formula and integration-by-parts. Here is an outline for a simplified case in 1d, where g is constant (see Winkler [2023] for full proof):

Proof.

$$dx = f(x)dt + gdw, \quad dw \sim \sqrt{dt}\mathcal{N}(0,1)$$

$$\text{For any } \phi\text{:} \quad d\phi(x) = \left(f(x)\partial_x\phi(x) + \frac{1}{2}g^2\partial_x^2\phi(x)\right)dt + g\partial_x\phi(x)dw \qquad \text{Itô's formula}$$

$$\implies \frac{d}{dt}\mathbb{E}[\phi] = \mathbb{E}[f\partial_x\phi + \frac{1}{2}g^2\partial_x^2\phi], \quad (\mathbb{E}[dw] = 0)$$

$$\int \phi(x)\partial_t p(x,t)dx = \int f(x)\partial_x\phi(x)p(x,t)dx + \frac{1}{2}g^2\int \partial_x^2\phi(x)p(x,t)dx$$

$$= -\int \phi(x)\partial_x(f(x)p(x,t))dx + \frac{1}{2}g^2\int \phi(x)\partial_x^2 p(x,t)dx, \qquad \text{integration-by-parts}$$

$$\partial_t p(x) = -\partial_x(f(x)p(x,t)) + \frac{1}{2}g^2\partial_x^2 p(x), \qquad \text{Fokker-Planck}$$

□

B.3 DDIM Point-mass Claim

Here is a version of Claim 3 where p_0 is a delta at an arbitrary point x_0.

Claim 5. *Suppose the target distribution is a point mass at $x_0 \in \mathbb{R}^d$, i.e. $p_0 = \delta_{x_0}$. Define the function*

$$G_t[x_0](x_t) = \left(\frac{\sigma_{t-\Delta t}}{\sigma_t}\right)(x_t - x_0) + x_0. \tag{115}$$

Then we clearly have $G_t[x_0]\sharp p_t = p_{t-\Delta t}$, and moreover

$$G_t[x_0](x_t) = x_t + \lambda(\mathbb{E}[x_{t-\Delta t} \mid x_t] - x_t) =: F_t(x_t). \tag{116}$$

Thus Algorithm 2 defines a valid reverse sampler for target distribution $p_0 = \delta_{x_0}$.

B.4 Flow Combining Lemma

Here we provide a more formal statement of the marginal flow result stated in Equation (64).

Equation (64) follows from a more general lemma (Lemma 3) which formalizes the "gas combination" analogy of Section 3. The motivation for this lemma is, we need a way of combining flows: of taking several different flows and producing a single "effective flow."

As a warm-up for the lemma, suppose we have n different flows, each with their own initial and final distributions q_i, p_i:

$$q_1 \overset{v^{(1)}}{\hookrightarrow} p_1, \quad q_2 \overset{v^{(2)}}{\hookrightarrow} p_2, \quad \ldots, \quad q_n \overset{v^{(n)}}{\hookrightarrow} p_n$$

We can imagine these as the flow of n different gases, where gas i has initial density q_i and final density p_i. Now we want to construct an overall flow v^* which takes the average initial-density to the average final-density:

$$\mathop{\mathbb{E}}_{i\in[n]}[q_i] \overset{v^*}{\hookrightarrow} \mathop{\mathbb{E}}_{i\in[n]}[p_i]. \tag{117}$$

To construct $v_t^*(x_t)$, we must take an average of the individual vector fields $v^{(i)}$, weighted by the probability mass the i-th flow places on x_t, at time t. (This is exactly analogous to Figure 5).

This construction is formalized in Lemma 3. There, instead of averaging over just a finite set of flows, we are allowed to average over any *distribution* over flows. To recover Equation (64), we can apply Lemma 3 to a distribution Γ over $(v, q_v) = (v^{[x_1,x_0]}, \delta_{x_1})$, that is, pointwise flows and their associated initial delta distributions.

Lemma 3 (Flow Combining Lemma)**.** *Let Γ be an arbitrary joint distribution over pairs (v, q_v) of flows v and their associated initial distributions q_v. Let $v(q_v)$ denote the final distribution when initial distribution q_v is transported by flow v, so $q_v \overset{v}{\hookrightarrow} v(q_v)$*

For fixed $t \in [0,1]$, consider the joint distribution over $(x_1, x_t, w_t) \in (\mathbb{R}^d)^3$ generated by:

$$\begin{aligned}(v, q_v) &\sim \Gamma \\ x_1 &\sim q_v \\ x_t &:= \text{RunFlow}(v, x_1, t) \\ w_t &:= v_t(x_t).\end{aligned}$$

Then, taking all expectations w.r.t. this joint distribution, the flow v^* defined as

$$\begin{aligned} v_t^*(x_t) &:= \mathbb{E}[w_t \mid x_t] && (118)\\ &= \mathbb{E}[v_t(x_t) \mid x_t] && (119)\end{aligned}$$

is known as the marginal flow for Γ*, and transports:*

$$\mathbb{E}[q_v] \overset{v^*}{\hookrightarrow} \mathbb{E}[v(q_v)]. \tag{120}$$

B.5 Derivation of DDIM using Flows

Now that we have the machinery of flows in hand, it is fairly easy to derive the DDIM algorithm "from scratch", by extending our simple scaling algorithm from the single point-mass case.

First, we need to find the pointwise flow. Recall from Claim 5 that for the simple case where the target distribution p_0 is a Dirac-delta at x_0, the following scaling maps p_t to $p_{t-\Delta t}$:

$$G_t[x_0](x_t) = \left(\frac{\sigma_{t-\Delta t}}{\sigma_t}\right)(x_t - x_0) + x_0 \implies G_t \sharp p_t = p_{t-\Delta t}.$$

G_t implies the pointwise flow:

$$\begin{aligned}
&\lim_{t\to 0} \left(\frac{\sigma_{t-\Delta t}}{\sigma_t}\right) = \sqrt{1-\frac{\Delta t}{t}} = (1-\frac{\Delta t}{2t}) \\
&\implies v_t^{[x_1,x_0]}(x_t) = -\lim_{\Delta t\to 0}\frac{G_t(x_t)-x_t}{\Delta t} = \frac{1}{2t}(x_t - x_0),
\end{aligned}$$

which agrees with (70).

Now let us compute the marginal flow v^* generated by the pointwise flow of Equation (70) and the coupling implied by the diffusion forward process. By Equation (69), the marginal flow is:

$$\begin{aligned}
v_t^*(x_t) &= \mathop{\mathbb{E}}_{x_1,x_0|x_t}[v_t^{[x_1,x_0]}(x_t) \mid x_t] && \text{By gas-lemma.} \\
&= \frac{1}{2t}\mathop{\mathbb{E}}_{\substack{x_0\sim p;\ x_1\leftarrow x_0+\mathcal{N}(0,\sigma_q^2) \\ x_t\leftarrow \text{RunFlow}(v_t^{[x_1,x_0]},x_1,t)}}[x_0 - x_t \mid x_t] && \text{For our choices of coupling and flow.} \\
&= \frac{1}{2t}\mathop{\mathbb{E}}_{\substack{x_0\sim p;\ x_1\leftarrow x_0+\mathcal{N}(0,\sigma_q^2) \\ x_t\leftarrow x_1\sqrt{t}+(1-\sqrt{t})x_0}}[x_0 - x_t \mid x_t] && \text{Expanding the flow trajectory.} \\
&= \frac{1}{2t}\mathop{\mathbb{E}}_{\substack{x_0\sim p \\ x_t\leftarrow\sqrt{t}\mathcal{N}(0,\sigma_q^2)}}[x_0 - x_t \mid x_t] && \text{Plugging in } x_1 = x_0+\mathcal{N}(0,\sigma_q^2).
\end{aligned}$$

This is exactly the differential equation describing the trajectory of DDIM (see Equation 58, which is the continuous-time limit of Equation 33).

B.6 Two Pointwise Flows for DDIM give the same Trajectory

We want to show that pointwise flow 71:

$$v_t^{[x_1,x_0]}(x_t) = \frac{1}{2\sqrt{t}}(x_0 - x_1) \tag{121}$$

is equivalent to the DDIM pointwise flow (70):

$$v_t^{[x_1,x_0]}(x_t) = \frac{1}{2t}(x_t - x_0) \tag{122}$$

because both these pointwise flows generate the same trajectory of x_t:

$$x_t = x_0 + (x_1 - x_0)\sqrt{t}. \tag{123}$$

To see this, we can solve the ODE determined by (70) via the Separable Equations method:

$$\begin{aligned}
\frac{dx_t}{dt} &= -\frac{1}{2t}(x_0 - x_t) \\
\implies \frac{\frac{dx_t}{dt}}{x_t - x_0} &= \frac{1}{2t} \\
\implies \int \frac{1}{x_t - x_0} dx &= \int \frac{1}{2t} dt, \text{ since } \frac{dx_t}{dt} dt = dx \\
\implies \log(x_t - x_0) &= \log\sqrt{t} + c \\
c &= \log(x_1 - x_0) \text{ (boundary cond.)} \\
\implies \log(x_t - x_0) &= \log\sqrt{t}(x_1 - x_0) \\
\implies x_t - x_0 &= \sqrt{t}(x_1 - x_0).
\end{aligned}$$

B.7 DDIM vs Time-reparameterized linear flows

Lemma 4 (DDIM vs Linear Flows)**.** *Let p_0 be an arbitrary target distribution. Let $\{x_t\}_t$ be the joint distribution defined by the DDPM forward process applied to p_0, so the marginal distribution of x_t is $p_t = p \star \mathcal{N}(0, t\sigma_q^2)$.*

Let $x^* \in \mathbb{R}^d$ be an arbitrary initial point. Consider the following two deterministic trajectories:

1. *The trajectory $\{y_t\}_t$ of the continuous-time DDIM flow, with respect to target distribution p_0, when started at initial point $y_1 = x^*$.*

   That is, y_t is the solution to the following ODE (Equation 58):

$$\frac{dy_t}{dt} = -v^{\text{ddim}}(y_t) \tag{124}$$

$$= -\frac{1}{2t} \mathop{\mathbb{E}}_{x_0|x_t} [x_0 - x_t \mid x_t = y_t] \tag{125}$$

   with boundary condition y_1 at $t = 1$.

2. *The trajectory $\{z_t\}_t$ produced when initial point $z_1 = x^*$ is transported by the marginal flow constructed from:*

   - *Linear pointwise flows*
   - *The DDPM-coupling of Line (73).*

That is, the marginal flow

$$\begin{aligned} v_t^\star(x_t) &= \mathop{\mathbb{E}}_{x_0,x_1|x_t}[v^{[x_1,x_0]}(x_t)|x_t] \\ &:= \mathop{\mathbb{E}}_{x_0,x_1|x_t}[x_0 - x_1|x_t] \\ &= \mathop{\mathbb{E}}_{x_0|x_t}[x_0 - x_t|x_t] \end{aligned}$$

since $\mathbb{E}[x_1|x_t] = x_t$ under the DDPM coupling.

Then, we claim these two trajectories are identical with the following time-reparameterization:

$$\forall t \in [0,1]: \quad y_t = z_{\sqrt{t}} \tag{126}$$

B.8 Proof Sketch of Claim 2

We will show that, in the forward diffusion setup of Section 1:

$$\mathbb{E}[(x_t - x_{t-\Delta t}) \mid x_t] = \frac{\Delta t}{t}\,\mathbb{E}[(x_t - x_0) \mid x_t]. \tag{127}$$

Proof sketch. Recall $\eta_t = x_{t+\Delta t} - x_t$. So by linearity of expectation:

$$\mathbb{E}[(x_t - x_0) \mid x_t] = \mathbb{E}[\sum_{i<t} \eta_i \mid x_t] \tag{128}$$

$$= \sum_{i<t} \mathbb{E}[\eta_i \mid x_t]. \tag{129}$$

Now, we claim that for given x_t, the conditional distributions $p(\eta_i \mid x_t)$ are identical for all $i < t$. To see this, notice that the joint distribution function $p(x_0, x_t, \eta_0, \eta_{\Delta t}, \ldots, \eta_{t-\Delta t})$ is symmetric in the $\{\eta_i\}$s, by definition of the forward process, and therefore the conditional distribution function $p(\eta_0, \eta_{\Delta t}, \ldots, \eta_{t-\Delta t} \mid x_t)$ is also symmetric in the $\{\eta_i\}$s. Therefore, all η_i have identical conditional expectations:

$$\mathbb{E}[\eta_0 \mid x_t] = \mathbb{E}[\eta_{\Delta t} \mid x_t] = \cdots = \mathbb{E}[\eta_{t-\Delta t} \mid x_t] \tag{130}$$

And since there are $(t/\Delta t)$ of them,

$$\sum_{i<t} \mathbb{E}[\eta_i \mid x_t] = \frac{t}{\Delta t}\,\mathbb{E}[\eta_{t-\Delta t} \mid x_t]. \tag{131}$$

Now continuing from Line 129,

$$\mathbb{E}[(x_0 - x_t) \mid x_t] = \sum_{i<t} \mathbb{E}[\eta_i \mid x_t] \tag{132}$$

$$= (t/\Delta t)\,\mathbb{E}[\eta_{t-\Delta t} \mid x_t] \tag{133}$$

$$= (t/\Delta t)\,\mathbb{E}[(x_t - x_{t-\Delta t}) \mid x_t] \tag{134}$$

as desired. □

B.9 Variance-Reduced Algorithms

Here we give the "varianced-reduced" versions of the DDPM training and sampling algorithms, where we train a network g_θ to approximate

$$g_\theta(x,t) \approx \mathbb{E}[x_0 \mid x_t] \tag{135}$$

instead of a network f_θ to approximate

$$f_\theta(x,t) \approx \mathbb{E}[x_{t-\Delta t} \mid x_t]. \tag{136}$$

Via Claim 2, these two functions are equivalent via the transform:

$$f_\theta(x,t) = (\Delta t/t) g_\theta(x,t) + (1-\Delta t/t)x. \tag{137}$$

Plugging this relation into Pseudocode 2 yields the variance-reduced DDPM sampler of Pseudocode 7.

Pseudocode 6: DDPM train loss (x_0-prediction)

Input: Neural network g_θ; Sample-access to train distribution p.
Data: Terminal variance σ_q
Output: Stochastic loss L

$x_0 \leftarrow \text{Sample}(p)$
$t \leftarrow \text{Unif}[0,1]$
$x_t \leftarrow x_0 + \mathcal{N}(0, \sigma_q^2 t)$
$L \leftarrow \|g_\theta(x_t,t) - x_0\|_2^2$
return L

Pseudocode 7: DDPM sampling (x_0-prediction)

Input: Trained model f_θ.
Data: Terminal variance σ_q; step-size Δt.
Output: x_0

$x_1 \leftarrow \mathcal{N}(0, \sigma_q^2)$
for $t = 1,\ (1-\Delta t), (1-2\Delta t), \ldots, \Delta t$ **do**
  $\widehat{\eta}_t \leftarrow g_\theta(x_t,t) - x_t$
  $x_{t-\Delta t} \leftarrow x_t + (1/t)\widehat{\eta}_t \Delta t + \mathcal{N}(0, \sigma_q^2 \Delta t)$
end
return x_0

B.10 Equivalence of and x_0- and ε-prediction

We will discuss this in our usual simplified setup:

$$x_t = x_0 + \sigma_t \varepsilon_t, \quad \sigma_t = \sigma_q \sqrt{t}, \quad \varepsilon_t \sim \mathcal{N}(0,1);$$

the scaling factors are more complex in the general case (see Luo [2022] for VP diffusion, for example) but the idea is the same. The DDPM training algorithm 1 has objective and optimal value

$$\min_\theta \quad \|f_\theta(x_t,t) - x_{t-\Delta t}\|_2^2, \quad f_\theta^\star(x_t,t) = \mathbb{E}[x_{t-\Delta t}|x_t]$$

That is, the network f_θ to learn to predict $\mathbb{E}[x_{t-\Delta t}|x_t]$. However, we could instead require the network to predict other related quantities, as follows. Noting that

$$\mathbb{E}[x_{t-\Delta t} - x_t | x_t] \overset{\text{eq. 23}}{=} \frac{\Delta t}{t}\mathbb{E}[x_0 - x_t|x_t] \equiv \frac{\Delta t}{t\sigma_t}\mathbb{E}[\varepsilon_t|x_t]$$

$$\implies \|E[x_{t-\Delta t} - x_t|x_t] - x_{t-\Delta t}\|_2^2 = \|\frac{\Delta t}{t}(\mathbb{E}[x_0|x_t] - x_0)\|_2^2 = \|\frac{\Delta t}{t\sigma_t}(\mathbb{E}[\varepsilon_t|x_t] - \varepsilon_t)\|_2^2$$

we get the following equivalent problems:

$$\min_\theta \quad \|f_\theta(x_t,t)-x_0\|_2^2 \implies f_\theta^\star(x_t,t)=\mathbb{E}[x_0|x_t], \quad \text{time-weighting } = \frac{1}{t}$$
$$\min_\theta \quad \|\frac{\Delta t}{t\sigma_t}(f_\theta(x_t,t)-\varepsilon_t)\|_2^2 \implies f_\theta^\star(x_t,t)=\mathbb{E}[\varepsilon_t|x_t] \quad \text{time-weighting } = \frac{1}{t\sigma_t}.$$

References

Madhu S Advani, Andrew M Saxe, and Haim Sompolinsky. High-dimensional dynamics of generalization error in neural networks. *Neural Networks*, 132:428–446, 2020. ↑34

Michael S. Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions, 2023. ↑25, ↑31, ↑35, ↑37

Michael Samuel Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *The Eleventh International Conference on Learning Representations*, 2022. ↑31

Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12 (3):313–326, 1982. ↑14

Nicolas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramer, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 5253–5270, 2023. ↑23

Stanley H. Chan. Tutorial on diffusion models for imaging and vision, 2024. ↑36

Hongrui Chen, Holden Lee, and Jianfeng Lu. Improved analysis of score-based generative modeling: User-friendly bounds under minimal smoothness assumptions. In *International Conference on Machine Learning*, pages 4735–4763. PMLR, 2023. ↑23

Sitan Chen, Sinho Chewi, Jerry Li, Yuanzhi Li, Adil Salim, and Anru Zhang. Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions. In *The Eleventh International Conference on Learning Representations*, 2022. ↑23

Sitan Chen, Sinho Chewi, Holden Lee, Yuanzhi Li, Jianfeng Lu, and Adil Salim. The probability flow ode is provably fast. *Advances in Neural Information Processing Systems*, 36, 2024a. ↑23

Sitan Chen, Vasilis Kontonis, and Kulin Shah. Learning general gaussian mixtures with efficient score matching. *arXiv preprint arXiv:2404.18893*, 2024b. ↑23

Ayan Das. Building diffusion model's theory from ground up. In *ICLR Blogposts 2024*, 2024. URL https://iclr-blogposts.github.io/2024/blog/diffusion-theory-from-scratch/. https://iclr-blogposts.github.io/2024/blog/diffusion-theory-from-scratch/. ↑36

Valentin De Bortoli. Convergence of denoising diffusion models under the manifold hypothesis. *arXiv preprint arXiv:2208.05314*, 2022. ↑23

Valentin De Bortoli, James Thornton, Jeremy Heng, and Arnaud Doucet. Diffusion schrödinger bridge with applications to score-based generative modeling. *Advances in Neural Information Processing Systems*, 34: 17695–17709, 2021. ↑23

Sander Dieleman. Perspectives on diffusion, 2023. URL https://sander.ai/2023/07/20/perspectives.html. ↑36

Tony Duan. Diffusion models from scratch, 2023. URL https://www.tonyduan.com/diffusion/index.html. ↑36

Ronen Eldan. Lecture notes - from stochastic calculus to geometric inequalities, 2024. URL https://www.wisdom.weizmann.ac.il/~ronene/GFANotes.pdf. ↑13

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. *arXiv preprint arXiv:2403.03206*, 2024. ↑25

Lawrence C Evans. *An introduction to stochastic differential equations*, volume 82. American Mathematical Soc., 2012. ↑13

Tor Fjelde, Emile Mathieu, and Vincent Dutordoir. An introduction to flow matching, January 2024. URL https://mlg.eng.cam.ac.uk/blog/2024/01/20/flow-matching.html. ↑31, ↑37

Xiangming Gu, Chao Du, Tianyu Pang, Chongxuan Li, Min Lin, and Ye Wang. On memorization in diffusion models. *arXiv preprint arXiv:2310.02664*, 2023. ↑23

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. ↑3, ↑8, ↑32, ↑33

Zahra Kadkhodaie, Florentin Guth, Eero P Simoncelli, and Stéphane Mallat. Generalization in diffusion models arises from geometry-adaptive harmonic representations. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ANvmVS2Yr0. ↑33

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models, 2022. ↑12, ↑14, ↑32, ↑36, ↑37

Diederik P Kingma and Ruiqi Gao. Understanding diffusion objectives as the ELBO with simple data augmentation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=NnMEadcdyD. ↑33

P.E. Kloeden and E. Platen. *Numerical Solution of Stochastic Differential Equations*. Stochastic Modelling and Applied Probability. Springer Berlin Heidelberg, 2011. ISBN 9783540540625. URL https://books.google.com/books?id=BCvtssom1CMC. ↑13

Holden Lee, Jianfeng Lu, and Yixin Tan. Convergence of score-based generative modeling for general data distributions. In *International Conference on Algorithmic Learning Theory*, pages 946–985. PMLR, 2023. ↑23

Shanchuan Lin, Anran Wang, and Xiao Yang. Sdxl-lightning: Progressive adversarial diffusion distillation. *arXiv preprint arXiv:2402.13929*, 2024. ↑32

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t. ↑25, ↑28

Xingchao Liu, Chengyue Gong, et al. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2022a. ↑25, ↑28, ↑29, ↑31

Xingchao Liu, Lemeng Wu, Mao Ye, and Qiang Liu. Let us build bridges: Understanding and extending diffusion generative models, 2022b. ↑25

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022a. ↑32

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022b. ↑32

Calvin Luo. Understanding diffusion models: A unified perspective, 2022. ↑33, ↑36, ↑46

David McAllester. On the mathematics of diffusion models, 2023. ↑36

Andrea Montanari. Sampling, diffusions, and stochastic localization, 2023. ↑37

Stefano Peluchetti. Non-denoising forward-time diffusions, 2022. URL https://openreview.net/forum?id=oVfIKuhqfC. ↑25

Frank Permenter and Chenyang Yuan. Interpreting and improving diffusion models using the euclidean distance function. *arXiv preprint arXiv:2306.04848*, 2023. ↑36

Gabriel Peyré. Denoising diffusion models, 2023. URL https://mathematical-tours.github.io/book-sources/optim-ml/OptimML-DiffusionModels.pdf. ↑37

Aram-Alexandre Pooladian, Heli Ben-Hamu, Carles Domingo-Enrich, Brandon Amos, Yaron Lipman, and Ricky TQ Chen. Multisample flow matching: Straightening flows with minibatch couplings. In *International Conference on Machine Learning*, pages 28100–28127. PMLR, 2023. ↑31

Fabio De Sousa Ribeiro and Ben Glocker. Demystifying variational diffusion models, 2024. ↑36

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022. ↑34

Axel Sauer, Frederic Boesel, Tim Dockhorn, Andreas Blattmann, Patrick Esser, and Robin Rombach. Fast high-resolution image synthesis with latent adversarial diffusion distillation. *arXiv preprint arXiv:2403.12015*, 2024. ↑32

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *CoRR*, abs/1503.03585, 2015. URL http://arxiv.org/abs/1503.03585. ↑4, ↑8, ↑10, ↑33, ↑36

Gowthami Somepalli, Vasu Singla, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Diffusion art or digital forgery? investigating data replication in diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6048–6058, 2023. ↑23

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=St1giarCHLP. ↑3, ↑16, ↑32

Jiaming Song, Chenlin Meng, and Arash Vahdat. Cvpr 2023 tutorial: Denoising diffusion models: A generative learning big bang, 2023a. URL https://cvpr2023-tutorial-diffusion-models.github.io. ↑36

Yang Song. Generative modeling by estimating gradients of the data distribution, 2021. URL https://yang-song.net/blog/2021/score/. ↑13

Yang Song. Diffusion and score-based generative models, 2023. URL https://www.youtube.com/watch?v=wMmqCMwuM2Q. ↑36

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019. ↑32

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. URL https://arxiv.org/pdf/2011.13456.pdf. ↑14, ↑21, ↑32, ↑37, ↑40

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023b. ↑32

Hannes Stark, Bowen Jing, Chenyu Wang, Gabriele Corso, Bonnie Berger, Regina Barzilay, and Tommi Jaakkola. Dirichlet flow matching with applications to dna sequence design, 2024. ↑31

Alexander Tong, Nikolay Malkin, Kilian Fatras, Lazar Atanackovic, Yanlei Zhang, Guillaume Huguet, Guy Wolf, and Yoshua Bengio. Simulation-free schr\" odinger bridges via score and flow matching. *arXiv preprint arXiv:2307.03672*, 2023. ↑31

Angus Turner. Diffusion models as a kind of vae, June 2021. URL https://angusturner.github.io/generative_models/2021/06/29/diffusion-probabilistic-models-I.html. ↑33

Ludwig Winkler. Reverse time stochastic differential equations [for generative modeling], 2021. URL https://ludwigwinkler.github.io/blog/ReverseTimeAnderson/. ↑14

Ludwig Winkler. Fokker, planck, and ito, 2023. URL https://ludwigwinkler.github.io/blog/FokkerPlanck/. ↑41

Yanwu Xu, Yang Zhao, Zhisheng Xiao, and Tingbo Hou. Ufogen: You forward once large scale text-to-image generation via diffusion gans. *arXiv preprint arXiv:2311.09257*, 2023. ↑32

Chenyang Yuan. Diffusion models from scratch, from a new theoretical perspective, 2024. URL https://www.chenyang.co/diffusion.html. ↑36

Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Loek7hfb46P. ↑32