Action Matching: A Variational Method for Learning Stochastic Dynamics from Samples

Kirill Neklyudov
Vector Institute

Daniel Severo
University of Toronto
Vector Institute

Alireza Makhzani
University of Toronto
Vector Institute

Abstract

Stochastic dynamics are ubiquitous in many fields of science, from the evolution of quantum systems in physics to diffusion-based models in machine learning. Existing methods such as score matching can be used to simulate these physical processes by assuming that the dynamics is a diffusion, which is not always the case. In this work, we propose a method called "Action Matching" that enables us to learn a much broader family of stochastic dynamics. Our method requires access only to samples from different time-steps, makes no explicit assumptions about the underlying dynamics, and can be applied even when samples are uncorrelated (i.e., are not part of a trajectory). Action Matching directly learns an underlying mechanism to move samples in time without modeling the distributions at each time-step. In this work, we showcase how Action Matching can be used for several computer vision tasks such as generative modeling, super-resolution, colorization, and inpainting; and further discuss potential applications in other areas of science.

1 Problem Formulation of Learning Continuous Dynamics

The problem of learning stochastic dynamics is one of the most fundamental problems in many different fields of science. In physics, porous medium equations (Vázquez, 2007) describe many natural phenomena from this perspective, such as Fokker Planck equation in statistical mechanics, Vlasov equation for plasma, and Nonlinear heat equation. Another prominent example is from Quantum Mechanics where the state of physical systems is a distribution whose evolution is described by the Schrödinger equation. Recently, stochastic dynamics have achieved very promising results in machine learning applications. The most promising examples of this approach are the diffusion-based generative models (Song et al., 2020b; Ho et al., 2020).

Informal Problem Setup In this paper, we approach the problem of *Learning Stochastic Dynamics* from their samples. Suppose we observe the time evolution of some random variable X_t with the density q_t, from t_0 to t_1. Having access to samples from the density q_t at different points in time $t \in [t_0, t_1]$, we want to build a model of the dynamics by learning how to move samples in time such that they respect the marginals q_t. In this work, we propose a method called "Action Matching" as a solution to this problem.

Continuity Equation Suppose we have a set of particles in space $\mathcal{X} \subset \mathbb{R}^d$, initially distributed as q_{t_0}. Let each particle follow a time-dependent ODE (continuous flow) with the velocity field $v : [t_0, t_1] \times \mathcal{X} \to \mathbb{R}^d$ as follows

$$\frac{\partial}{\partial t} x(t) = v_t(x(t)) \,, \quad x(t_0) = x \,. \tag{1}$$

Correspondence to k.necludov@gmail.com, makhzani@vectorinstitute.ai. See arXiv for the extended paper.

NeurIPS 2022 Workshop on Score-Based Methods.

From fluid mechanics, we know that the density of the particles at time t, denoted by q_t, evolves according to the *continuity equation*

$$\frac{\partial}{\partial t} q_t = -\nabla \cdot (q_t v_t)\,, \tag{2}$$

which holds in the distributional sense, where $\nabla\cdot$ denotes the divergence operator.

Note that even though we arrived at the continuity equation using ODEs, the continuity equation can describe a rich family of density evolutions in a wide range of stochastic processes, including those of SDEs (see Equation 37 of Song et al. (2020b)), or even those of the porous medium equation (Otto, 2001) that are more general than SDEs. This motivates us to restrict ourselves to ODEs of the form Eq. (1), and the continuity equation, without losing any modelling capacity. In fact, as the following theorem shows, under mild conditions, *any continuous dynamics can be modeled by the continuity equation*, and moreover any continuity equation results in a continuous dynamics.

Theorem 1 (Adapted from Theorem 8.3.1 of Ambrosio et al. (2008))**.** *Consider a continuous dynamic with the density evolution of q_t, which satisfies mild conditions (absolute continuity in the 2-Wasserstein space of distributions $\mathcal{P}_2(\mathcal{X})$). Then, there exists a unique (up to a constant) function $s_t^*(x)$, called "action", such that vector field $v_t^*(x) = \nabla s_t^*(x)$ and q_t satisfies the continuity equation*

$$\frac{\partial}{\partial t} q_t = -\nabla \cdot (q_t \nabla s_t^*(x))\,. \tag{3}$$

In other words, the ODE $\frac{\partial}{\partial t} x(t) = \nabla s_t^*(x)$ can be used to move samples in time such that the marginals are q_t.

Using Theorem 1, the problem of learning the dynamics can be boiled down to learning the unique vector field ∇s_t^*, only using samples from q_t. Motivated by this, we restrict our search space of velocity vectors to the family of gradient vector fields

$$\mathcal{S}_t = \{\nabla s_t \,|\, s_t : \mathcal{X} \to \mathbb{R}\}\,. \tag{4}$$

We use a neural network to parameterize the set of functions $s_t(x)$, and propose Action Matching for learning the neural network such that $s_t(x)$ approximates $s_t^*(x)$. Once we have learned the vector field ∇s_t^*, we can move samples forward or backward in time by simulating the ODE in Eq. (1) with the velocity ∇s_t^*. The continuity equation ensures that samples at any given time $t \in [t_0, t_1]$ are distributed according to q_t.

2 Action Matching

The main development of this paper is the Action Matching method, which allows us to recover the *true action* s_t^* of a continuous dynamic and thereby simulate it, while having access only to samples from q_t. In order to do so, we define the *variational action* $s_t(x)$, parameterized by a neural network, that approximates $s_t^*(x)$, by minimizing the "ACTION-GAP" objective

$$\text{ACTION-GAP}(s, s^*) \coloneqq \frac{1}{2} \int \mathbb{E}_{q_t(x)} \|\nabla s_t(x) - \nabla s_t^*(x)\|^2 dt\,. \tag{5}$$

Note that this objective is intractable, as we do not have access to ∇s^*. We now propose action matching as a variational framework for optimizing this objective.

Proposition 1. *For an arbitrary variational action s, the gap can be characterized as*

$$\text{ACTION-GAP}(s, s^*) = \mathrm{L}(s) + \frac{1}{2} \int_{t_0}^{t_1} \mathbb{E}_{q_t(x)} \|\nabla s_t^*(x)\|^2 dt\,, \tag{6}$$

where

$$\mathrm{L}(s) = \mathbb{E}_{q_{t_0}(x)}[s_{t_0}(x)] - \mathbb{E}_{q_{t_1}(x)}[s_{t_1}(x)] + \int \mathbb{E}_{q_t(x)} \left[\frac{1}{2}\|\nabla s_t(x)\|^2 + \frac{\partial s_t}{\partial t}(x)\right] dt\,. \tag{7}$$

Thus, the following optimization problems are equivalent

$$s^* = \arg\min_s \mathrm{L}(s) = \arg\min_s \text{ACTION-GAP}(s, s^*)\,, \tag{8}$$

where the equality is up to an additive constant, and the minimum is achieved iff $\nabla s_t(x) = \nabla s_t^*(x)$.

See Appendix A for the proof. Proposition 1 indicates that minimizing Eq. (7) results in minimizing the the ACTION-GAP from Eq. (5). However, unlike the intractable ACTION-GAP, minimising L is tractable, as we can use the samples of q_t to obtain its unbiased low variance estimate.

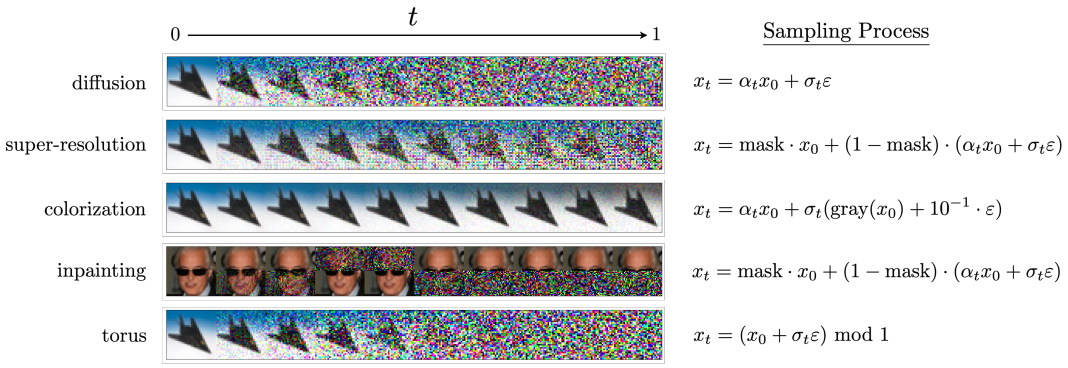

Figure 1: Examples of different noising processes used for different vision tasks. At $t = 0$, we start from the data distribution. Depending on the task, the noising process gradually destroys all or partial information of data, and replace it with prior noise.

3 Generative Modeling using Action Matching

Action Matching has a wide range of applications in generative modeling. In Action Matching generative models, we first have to define a dynamics (i.e., noising process) that transforms samples from the data distribution $q_0 = \pi$ to samples of a prior distribution q_1 (e.g., standard Gaussian). Action Matching is then used to learn the vector field $\nabla s^\star$ of the chosen dynamics. Once $\nabla s^\star$ is learned, we can sample from the target distribution by first sampling from the prior, and then moving the samples using a reverse ODE with the velocity $\nabla s^\star$. Finally, Action Matching enables us to compute the exact log-likelihood of the data.

Noising Processes in Action Matching Generative Models To learn the vector field $\nabla s^\star$, Action Matching only requires samples from the intermediate distributions $q_t, t \in [0, 1]$, that define the noising process. We now provide a broad family of noising processes that can be used for generative modeling tasks. Consider the process

$$x_t = f_t(x_0) + \sigma_t \varepsilon, \quad x_0 \sim \pi(x), \quad \varepsilon \sim p(\varepsilon)\,, \tag{9}$$

where $f_t(x_0)$ is some transformation of the data, which could be nonlinear. At $t = 0$, f_0 is the identity function, and $\sigma_0 = 0$. Thus, x_0 is distributed according to the data distribution, i.e., $q_0(x_0) = \pi(x_0)$. The noising process then gradually eliminates information from the samples using f_t, and increases the variance of noise σ_t. At $t = 1$, f_t would become the zero function and we have $\sigma_1 = 1$. Thus, x_1 would be distributed as $q_1(x_1) = p(x_1)$. See Fig. 1 for the examples of these sampling processes. We will demonstrate Action Matching learning these dynamics in the experiment section.

Learning Once we define the noising process for $q_t, \ \forall t \in [0, 1]$, we apply Action Matching. It samples points with different time-steps and then minimizes the objective (5) w.r.t. the parameters θ of $s_t(x, \theta)$. In practice, we reduce the variance of the objective (5), by weighting it over time and adaptively selecting the distribution of sampled time-steps. We derive the *weighted* objective in Appendix A, and further discuss the details of training in Appendix B.

Sampling We sample from the target distribution via the trained function $s_t(x(t), \theta^*)$ by solving the following ODE backward in time:

$$\frac{\partial}{\partial t} x = \nabla_x s_t(x(t), \theta^*), \quad x(t = 1) = \varepsilon\,, \quad \varepsilon \sim p(\varepsilon). \tag{10}$$

Recall that this sampling process is justified by Eq. (3), where $s_t(x(t), \theta^*)$ approximates s_t^*.

Evaluating the Log-likelihood for the generation tasks can be done by integrating the same ODE forward, i.e.,

$$\log q_0(x(0)) = \log q_1(x(1)) + \int_0^1 dt\ \nabla^2 s_t^*(x(t)), \quad \frac{\partial}{\partial t} x = \nabla_x s_t^*(x(t)), \quad x(t = 0) = x, \tag{11}$$

where we approximate s_t^* by $s_t(x(t), \theta^*)$ and assume the density $q_1(x)$ to be a known analytic distribution.

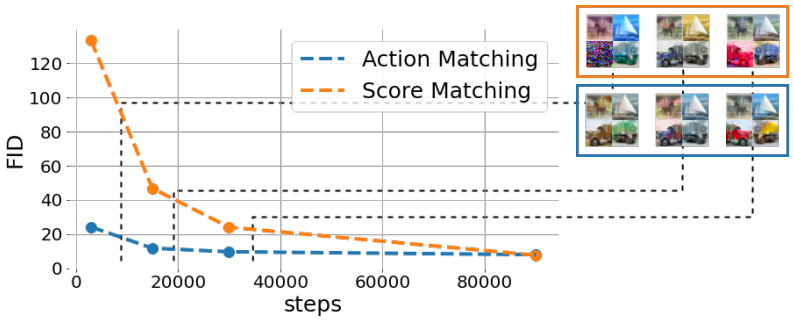

Figure 2: Faster convergence of Action Matching (AM) compared to Score Matching (SM) in FID values and generated samples quality for the colorization task on CIFAR-10.

Method	Average MMD↓
AM (ours)	$5.7 \cdot 10^{-4} \pm 3.1 \cdot 10^{-4}$
ALD + SM	$4.8 \cdot 10^{-2} \pm 4.8 \cdot 10^{-3}$
ALD + SSM	$4.7 \cdot 10^{-2} \pm 4.0 \cdot 10^{-3}$
ALD + True Scores	$3.6 \cdot 10^{-2} \pm 4.1 \cdot 10^{-4}$

Table 1: Performance of Action Matching and the Annealed Langevin Dynamics (ALD) for the Schrödinger equation simulation. For ALD, we estimate the scores in two ways: Score Matching (SM) and Sliced Score Matching (SSM). We also demonstrate that even using true scores does not allow for the precise simulation.

4 Experiments

Generative modeling We apply Action Matching to MNIST, CelebA (Liu et al., 2015) and CIFAR-10 datasets for a variety of computer vision tasks. Namely, we perform unconditional image generation via diffusion as well as conditional generation for super-resolution, in-painting, and colorization tasks. In addition to these settings, we also learn unconditional image generation on a torus, where Denoising Score Matching can not be applied in the original formulation. We train all models for 300k iterations and report the negative log-likelihood in bits per dimension (BPD) and FID scores (Heusel et al., 2017) in Table 2. We illustrate the generative process in github.com/action-matching.

We observe that Denoising Score Matching performs better than Action Matching on all tasks due to the additional information that the Denoising Score Matching objective uses about the underlying process. However, we expect Action Matching to converge faster on the conditional image generation tasks, as it only needs to learn a cross-domain transformation, rather than learning the conditional generation from the Gaussian noise. We experimentally verified this hypothesis by evaluating the FID throughout the training process, on the colorization task, shown in Fig. 2.

Schrödinger equation simulation We demonstrate that Action Matching can learn a wide range of stochastic dynamics by applying it to the dynamics of a quantum system evolving according to the Schrödinger equation. Here, for the ground truth dynamics, we take the dynamics of an excited state of the hydrogen atom, which is described by the following equation

$$i\frac{\partial}{\partial t}\psi(x,t) = -\frac{1}{\|x\|}\psi(x,t) - \frac{1}{2}\nabla^2\psi(x,t). \tag{12}$$

The function $\psi(x,t) : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{C}$ is called a wavefunction and it completely defines the distribution of the coordinates x by defining its density as $q_t(x) := |\psi(x,t)|^2$.

For the baseline, we take Annealed Langevin Dynamics as considered in (Song & Ermon, 2019). It approximates the ground truth dynamics using only scores of the distributions by running the approximate MCMC method (which does not have access to the densities) targeting the intermediate distributions of the dynamics (see Algorithm 2). For the estimation of scores, we consider Score Matching (SM) (Hyvärinen & Dayan, 2005), Sliced Score Matching (SSM) (Song et al., 2020a), and additionally evaluate the baseline using the ground truth scores. For further details, we refer the reader to Appendix D.2 and the code github.com/action-matching.

Action Matching outperforms both Score Matching and Sliced Score Matching, precisely simulating the true dynamics (see Table 1). Note, that even using the ground truth scores in Annealed Langevin Dynamics does not match the performance of Action Matching (see Table 1) since it is itself an approximation to the Metropolis-Adjusted Langevin Algorithm. Finally, we provide animations of the learned dynamics for different methods (see github.com/action-matching) to illustrate the performance difference.

References

Luigi Ambrosio, Nicola Gigli, and Giuseppe Savaré. *Gradient flows: in metric spaces and in the space of probability measures*. Springer Science & Business Media, 2008.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

David J Griffiths and Darrell F Schroeter. *Introduction to quantum mechanics*. Cambridge university press, 2018.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pp. 3730–3738, 2015.

Felix Otto. The geometry of dissipative evolution equations: the porous medium equation. 2001.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Tim Salimans and Jonathan Ho. Should ebms model the energy or the score? In *Energy Based Models Workshop-ICLR 2021*, 2021.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation. In *Uncertainty in Artificial Intelligence*, pp. 574–584. PMLR, 2020a.

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*, 2020b.

Juan Luis Vázquez. *The porous medium equation: mathematical theory*. Oxford University Press on Demand, 2007.

A Action Matching

Proposition. *For an arbitrary variational action s, the gap can be characterized as*

$$\text{ACTION-GAP}(s, s^*) = \mathrm{L}(s) + \frac{1}{2}\int_{t_0}^{t_1} \mathbb{E}_{q_t(x)} \|\nabla s_t^*(x)\|^2 dt\,, \tag{13}$$

where

$$\mathrm{L}(s) = \underbrace{\mathbb{E}_{q_{t_0}(x)}[s_{t_0}(x)] - \mathbb{E}_{q_{t_1}(x)}[s_{t_1}(x)]}_{\textit{action-increment}} + \underbrace{\int \mathbb{E}_{q_t(x)}\left[\frac{1}{2}\|\nabla s_t(x)\|^2 + \frac{\partial s_t}{\partial t}(x)\right] dt}_{\textit{smoothness (regularization)}}\,. \tag{14}$$

Thus, the following optimization problems are equivalent

$$\arg\min_s \{\mathrm{L}(s)\} = \arg\min_s \{\text{ACTION-GAP}(s, s^*)\}\,, \tag{15}$$

where the equality is up to an additive constant, and the minimum is achieved iff $\nabla s_t(x) = \nabla s_t^*(x)$.

Proof.

$$\begin{aligned}
&\text{ACTION-GAP}(s, s^*) \\
&= \frac{1}{2}\int_{t_0}^{t_1} \omega_t \mathbb{E}_{q_t(x)} \|\nabla s - \nabla s^*\|^2 dt \\
&= \frac{1}{2}\int_{t_0}^{t_1}\int_x \omega_t q_t(x) \|\nabla s - \nabla s^*\|^2 dx dt \\
&= \frac{1}{2}\int_{t_0}^{t_1}\int_x \omega_t q_t(x)\|\nabla s\|^2 dx dt - \int_{t_0}^{t_1}\omega_t \int_x q_t(x)\langle \nabla s_t(x), \nabla s_t^*(x)\rangle dx dt + \overbrace{\frac{1}{2}\int \mathbb{E}_{q_t(x)}\|\nabla s^*\|^2 dt}^{\mathcal{K}} \\
&= \frac{1}{2}\int_{t_0}^{t_1}\int_x \omega_t q_t(x)\|\nabla s\|^2 dx dt - \int_{t_0}^{t_1}\omega_t \int_x \langle \nabla s_t(x), q_t(x)\nabla s_t^*(x)\rangle dx dt + \mathcal{K} \\
&\overset{(1)}{=} \frac{1}{2}\int_{t_0}^{t_1}\int_x \omega_t q_t(x)\|\nabla s\|^2 dx dt + \int_{t_0}^{t_1}\omega_t \int_x s_t(x)[\nabla\cdot(q_t(x)\nabla s_t^*(x))] dx dt + \mathcal{K} \\
&= \frac{1}{2}\int_{t_0}^{t_1}\int_x \omega_t q_t(x)\|\nabla s\|^2 dx dt - \int_{t_0}^{t_1}\left(\int_x \omega_t s_t(x)\frac{\partial q_t(x)}{\partial t} dx\right) dt + \mathcal{K} \\
&\overset{(2)}{=} \int_{t_0}^{t_1}\omega_t \mathbb{E}_{q_t(x)}\left[\frac{1}{2}\|\nabla s_t(x)\|^2\right] dt - \left(\omega_t \mathbb{E}_{q_t(x)}[s_t(x)]\Big|_{t_0}^{t_1} - \int_x \mathbb{E}_{q_t(x)}\left[s_t(x)\frac{d\omega_t}{dt} + \omega_t\frac{\partial s_t(x)}{\partial t}\right] dt\right) + \mathcal{K} \\
&= \int_{t_0}^{t_1}\omega_t \mathbb{E}_{q_t(x)}\left[\frac{1}{2}\|\nabla s_t(x)\|^2 + \frac{\partial s_t(x)}{\partial t} + s_t(x)\frac{d\log\omega_t}{dt}\right] dt - \omega_{t_1}\mathbb{E}_{q_{t_1}(x)}[s_{t_1}(x)] + \omega_{t_0}\mathbb{E}_{q_{t_0}(x)}[s_{t_0}(x)] + \mathcal{K} \\
&= \mathrm{L}(s) + \mathcal{K}\,,
\end{aligned}$$

where in (1), we have used $\int_V \langle \nabla g, \boldsymbol{f}\rangle dx = \oint_{\partial V}\langle g\boldsymbol{f}, \boldsymbol{ds}\rangle - \int_V g(\nabla\cdot\boldsymbol{f})dx$, and in (2) we have used integration by parts. □

B Generative Modeling in Practice

In practice, we found that the naive application of Action Matching for complicated dynamics such as image generation might exhibit poor convergence due to the large variance of objective estimate. Moreover, the optimization problem

$$\min_{s_t} \frac{1}{2}\int \mathbb{E}_{q_t(x)}\|\nabla s_t(x) - \nabla s_t^*(x)\|^2 dt \tag{16}$$

might be ill posed due to the singularity of the ground truth vector field ∇s_t^*. This happens when the data distribution q_0 is concentrated close to a low dimensional manifold, and the final distribution q_1 has a much higher intrinsic dimensionality (e.g., Gaussian distributions). In this case, the deterministic velocity vector field must be very large (infinite in the limit), so that it can pull apart the low dimensional manifold to transform it to higher dimensions.

Algorithm 1 Generative Modeling using Action Matching (In Practice)

Require: dataset $\{x^i\}_{i=1}^N$, $x^i \sim \pi(x) = q_0(x)$, batch-size n
Require: parameteric model $s_t(x, \theta)$, weight schedule $\omega(t)$
  for learning iterations **do**
    sample a mini-batch of data $\{x_0^i\}_{i=1}^n$ from the dataset $\{x^i\}_{i=1}^N$
    sample a mini-batch of noise $\{\varepsilon^i\}_{i=1}^n \sim q_1(x_1)$
    sample times $\{t^i\}_{i=1}^n \sim p(t)$
    sample two batches $\{x_1^i\}_{i=1}^n, \{x_{t^i}^i\}_{i=1}^n$ using $x_{t^i}^i = f_{t^i}(x_0^i) + \sigma_{t^i}\varepsilon^i$
    $\mathrm{L}_i = \left[s_0(x_0^i)\omega(0) - s_1(x_1^i)\omega(1) + \frac{1}{2}\left\|\nabla s_t(x_{t^i}^i)\right\|^2 \omega(t^i) + \frac{\partial s_t(x_{t^i}^i)}{\partial t}\omega(t^i) + s_t(x_t^i)\frac{\partial \omega(t^i)}{\partial t^i} \right]$
    $\mathrm{L} = \sum_{i=1}^n \frac{1}{p(t^i)}\mathrm{L}_i$
    update the model $\theta \leftarrow \text{Optimizer}(\theta, \nabla_\theta \mathrm{L}_\theta)$
  end for
  return trained model $s_t(x, \theta^*)$

We now discuss an example of this behavior, when the data distribution is a mixture of delta functions. Consider the sampling process

$$x_t = f_t(x_0) + \sigma_t \varepsilon, \quad x_0 \sim \pi(x), \quad \varepsilon \sim \mathcal{N}(x \,|\, 0, 1)\,, \tag{17}$$

where the target distribution is a mixture of delta-functions

$$\pi(x) = \frac{1}{N}\sum_i^N \delta(x - x^i). \tag{18}$$

Denoting the distribution of x_t as $q_t(x)$, we can solve the continuity equation

$$\frac{\partial q_t}{\partial t} = -\nabla \cdot (q_t v_t) \tag{19}$$

analytically (see Appendix C) by finding one of the many possible solutions

$$v_t = \frac{1}{\sum_i q_t^i(x)} \sum_i q_t^i(x) \left[(x - f_t(x^i)) \frac{\partial}{\partial t} \log \sigma_t + \frac{\partial f_t(x^i)}{\partial t} \right], \quad q_t^i(x) = \mathcal{N}(x \,|\, f_t(x^i), \sigma_t^2). \tag{20}$$

Note that v_t is not a gradient field in general, and thus is not the solution of action matching. However, it can be written as

$$v_t(x) = \sum_i \frac{q_t^i(x)}{\sum_i q_t^i(x)} \nabla s_t^i(x), \quad \text{where} \quad s_t^i(x) = \frac{1}{2}(x - f_t(x^i))^2 \frac{\partial}{\partial t} \log \sigma_t + \left\langle \frac{\partial f_t(x^i)}{\partial t}, x \right\rangle.$$

Given that the density of Gaussian distributions drop exponentially fast, we can conclude that for small values of t around each x^i, $\frac{q_t^j(x)}{\sum_j q_t^j(x)}$ is close to 1 if $i = j$, and close to 0 if $i \neq j$. Thus, $v_t(x)$ around each x^i can be locally approximated with the gradient vector field $\nabla s_t^i(x)$. Now suppose $\nabla s_t^*(x)$ is the solution of action matching, i.e., the unique gradient vector field that solves the continuity equation in every region, including regions around each x^i. Given the uniqueness of gradient vector fields that solve continuity equation, we can conclude that $\nabla s_t^i(x)$ locally matches $\nabla s_t^*(x)$ around each x^i.

For generative modeling, it is essential that $q_0 = \pi(x)$; hence, $\lim_{t\to 0} \sigma_t = 0$ and $\lim_{t\to 0} f_t(x) = x$. Assuming that σ_t^2 is continuous and differentiable at 0, in the limit, around each x^i, we have

$$\text{for } t \to 0\,, \quad \|\nabla s_t^*(x)\|^2 \propto \frac{1}{\sigma_t^2}, \quad \text{and} \quad \frac{1}{2}\mathbb{E}_{q_t(x)}\|\nabla s_t^*(x)\|^2 \propto \frac{1}{\sigma_t^2}. \tag{21}$$

Thus, the loss can be properly defined only on the interval $t \in (\delta, 1]$, where $\delta > 0$. In practice, we want to set δ as small as possible, i.e., we ideally want to learn s_t on the whole interval $t \in [0, 1]$. We can prevent learning the singularity functions just by re-weighting the objective in time as follows

$$\frac{1}{2}\int \mathbb{E}_{q_t(x)}\|\nabla s_t(x) - \nabla s_t^*(x)\|^2 dt \longrightarrow \frac{1}{2}\int \omega(t)\mathbb{E}_{q_t(x)}\|\nabla s_t(x) - \nabla s_t^*(x)\|^2 dt. \tag{22}$$

To give an example, we can take $\sigma_t = \sqrt{t}$ and $f_t(x) = x\sqrt{1-t}$, then $\omega(t) = (1-t)t^{3/2}$ cancels out the singularities at $t = 0$ and $t = 1$.

The second modification of the original Algorithm is the sampling of time-steps for the estimation of the time integral. Namely, the optimization of Equation (22) is equivalent to the minimization of the following objective

$$\mathrm{L}(s) = \underbrace{\omega(t_0)\mathbb{E}_{q_{t_0}(x)}[s_{t_0}(x)] - \omega(t_1)\mathbb{E}_{q_{t_1}(x)}[s_{t_1}(x)]}_{\text{weighted action-increment}} \tag{23}$$

$$+ \underbrace{\int_{t_0}^{t_1} \mathbb{E}_{q_t(x)}\left[\frac{1}{2}\omega(t)\|\nabla s_t(x)\|^2 + \omega(t)\frac{\partial s_t(x)}{\partial t} + s_t(x)\frac{\partial \omega(t)}{\partial t}\right] dt}_{\text{weighted smoothness}}, \tag{24}$$

which consists of two terms. Estimation of the weighted action-increment involves only sampling from q_{t_0} and q_{t_1}, while the weighted smoothness term estimate depends on the distribution of time samples $p(t)$, i.e.,

$$\int_{t_0}^{t_1} \underbrace{\frac{p(t)}{p(t)}}_{=1} \mathbb{E}_{q_t(x)}\left[\frac{1}{2}\omega(t)\|\nabla s_t(x)\|^2 + \omega(t)\frac{\partial s_t(x)}{\partial t} + s_t(x)\frac{\partial \omega(t)}{\partial t}\right] dt \tag{25}$$

$$= \mathbb{E}_{t\sim p(t)}\mathbb{E}_{x\sim q_t(x)}\frac{1}{p(t)}\left[\frac{1}{2}\omega(t)\|\nabla s_t(x)\|^2 + \omega(t)\frac{\partial s_t(x)}{\partial t} + s_t(x)\frac{\partial \omega(t)}{\partial t}\right]. \tag{26}$$

Note that $p(t)$ can be viewed as a proposal importance sampling distribution, and thus every choice of it results in an unbiased estimate of the original objective function. Thus, we can design $p(t)$ to reduce the variance of the weighted smoothness term of the objective. In our experiments, we observed that simply taking $p(t)$ proportionally to the standard deviation of the corresponding integrand significantly reduces the variance, i.e.,

$$p(t) \propto \sqrt{\mathbb{E}_{x\sim q_t}(\zeta_t - \mathbb{E}_{x\sim q_t}\zeta_t)^2}, \quad \zeta_t = \frac{1}{2}\omega(t)\|\nabla s_t(x)\|^2 + \omega(t)\frac{\partial s_t(x)}{\partial t} + s_t(x)\frac{\partial \omega(t)}{\partial t}. \tag{27}$$

We implement sampling from this distribution by aggregating the estimated variances throughout the training with exponential moving average, and then followed by linear interpolation between the estimates.

C Sparse Data Regime

In this section, we find velocity vector fields that satisfy the continuity equation in the case where the data distribution q_0 is a delta function or a mixture of delta functions; and the conditional $k_t(x_t \mid x)$ is a Gaussian distribution.

C.1 Delta Function Data Distribution

We start with the case where the dataset consists only of a single point $x_0 \in \mathbb{R}^d$

$$q_0(x) = \delta(x - x_0), \quad k_t(x_t \mid x) = \mathcal{N}(x_t \mid f_t(x), \sigma_t^2). \tag{28}$$

Then the distribution at time t is

$$q_t(x) = \int dx'\; q_0(x')k_t(x \mid x') = \mathcal{N}(x \mid f_t(x_0), \sigma_t^2). \tag{29}$$

The ground truth vector field v comes from the continuity equation

$$\frac{\partial q_t}{\partial t} = -\nabla \cdot (q_t v) \quad \Longrightarrow \quad \frac{\partial}{\partial t}\log q_t = -\langle \nabla \log q_t, v\rangle - \nabla \cdot (v). \tag{30}$$

For our dynamics, we have

$$\frac{\partial}{\partial t}\log q_t = \frac{\partial}{\partial t}\Big[-\frac{d}{2}\log(2\pi\sigma_t^2) - \frac{1}{2\sigma_t^2}\|x - f_t(x_0)\|^2\Big] \tag{31}$$

$$= -d\frac{\partial}{\partial t}\log\sigma_t + \frac{1}{\sigma_t^2}\|x - f_t(x_0)\|^2\frac{\partial}{\partial t}\log\sigma_t + \frac{1}{\sigma_t^2}\Big\langle x - f_t(x_0), \frac{\partial f_t(x_0)}{\partial t}\Big\rangle \tag{32}$$

$$= -d\frac{\partial}{\partial t}\log\sigma_t + \frac{1}{\sigma_t^2}\Big\langle x - f_t(x_0), (x - f_t(x_0))\frac{\partial}{\partial t}\log\sigma_t + \frac{\partial f_t(x_0)}{\partial t}\Big\rangle; \tag{33}$$

$$\nabla\log q_t = -\frac{1}{\sigma_t^2}(x - f_t(x_0)); \tag{34}$$

$$\frac{\partial}{\partial t}\log q_t = -d\frac{\partial}{\partial t}\log\sigma_t - \Big\langle\nabla\log q_t, (x - f_t(x_0))\frac{\partial}{\partial t}\log\sigma_t + \frac{\partial f_t(x_0)}{\partial t}\Big\rangle. \tag{35}$$

Matching the corresponding terms in the continuity equation, we get

$$v = (x - f_t(x_0))\frac{\partial}{\partial t}\log\sigma_t + \frac{\partial f_t(x_0)}{\partial t}. \tag{36}$$

We note that since the above vector field is a gradient flow, it is the unique vector field that the action matching would recover.

C.2 Mixture of Delta Functions Data Distribution

For the mixture of delta-functions, we denote

$$q_0(x) = \frac{1}{N}\sum_i^N \delta(x - x^i), \quad q_t(x) = \frac{1}{N}\sum_i^N q_t^i(x), \quad q_t^i(x) = \mathcal{N}(x \mid f_t(x^i), \sigma_t^2). \tag{37}$$

Due to the linearity of the continuity equation w.r.t. q, we have

$$\sum_i \frac{\partial q_t^i}{\partial t} = \sum_i \nabla\cdot\big(q_t^i v\big) \implies \sum_i q_t^i\Big(\frac{\partial}{\partial t}\log q_t^i + \langle\nabla\log q_t^i, v\rangle + \nabla\cdot(v)\Big) = 0. \tag{38}$$

We first solve the equation for $\frac{\partial f_t}{\partial t} = 0$, then for $\frac{\partial}{\partial t}\log\sigma_t = 0$ and join the solutions.

For $\frac{\partial f_t}{\partial t} = 0$, we look for the solution in the following form

$$v_\sigma = \frac{A}{\sum_i q_t^i}\sum_i \nabla q_t^i, \quad q_t^i(x) = \mathcal{N}(x \mid f_t^i(x^i), \sigma_t^2). \tag{39}$$

Then we have

$$\nabla\cdot(v_\sigma) = \Big\langle\nabla\frac{A}{\sum_i q_t^i}, \sum_i\nabla q_t^i\Big\rangle + \frac{A}{\sum_i q_t^i}\sum_i\nabla^2 q_t^i \tag{40}$$

$$= -\frac{A}{(\sum_i q_t^i)^2}\Big\|\sum_i\nabla q_t^i\Big\|^2 + \frac{A}{\sum_i q_t^i}\sum_i q_t^i\Big[\big\|\nabla\log q_t^i\big\|^2 - \frac{d}{\sigma_t^2}\Big], \tag{41}$$

$$\big(\sum_i q_t^i\big)\nabla\cdot(v_\sigma) = -\frac{A}{\sum_i q_t^i}\Big\|\sum_i\nabla q_t^i\Big\|^2 + A\sum_i q_t^i\Big[\big\|\nabla\log q_t^i\big\|^2 - \frac{d}{\sigma_t^2}\Big], \tag{42}$$

and from (38) we have

$$\sum_i q_t^i\Big(-d\frac{\partial}{\partial t}\log\sigma_t + \Big\langle\nabla\log q_t^i, v_\sigma + \sigma_t^2\frac{\partial}{\partial t}\log\sigma_t\nabla\log q_t^i\Big\rangle + \nabla\cdot(v_\sigma)\Big) = 0. \tag{43}$$

From these two equations we have

$$\sum_i q_t^i\nabla\cdot(v_\sigma) = -\frac{A}{\sum_i q_t^i}\Big\|\sum_i\nabla q_t^i\Big\|^2 + A\sum_i q_t^i\Big[\big\|\nabla\log q_t^i\big\|^2 - \frac{d}{\sigma_t^2}\Big] = \tag{44}$$

$$= \sum_i q_t^i\Big(d\frac{\partial}{\partial t}\log\sigma_t\Big) - \frac{A}{\sum_i q_t^i}\Big\|\sum_i\nabla q_t^i\Big\|^2 - \sigma_t^2\frac{\partial}{\partial t}\log\sigma_t\sum_i q_t^i\big\|\nabla\log q_t^i\big\|^2. \tag{45}$$

Thus, we have

$$A = -\sigma_t^2 \frac{\partial}{\partial t} \log \sigma_t. \tag{46}$$

For $\frac{\partial}{\partial t} \log \sigma_t = 0$, we simply check that the solution is

$$v_f = \frac{1}{\sum_i q_t^i} \sum_i q_t^i \frac{\partial f_t(x^i)}{\partial t}. \tag{47}$$

Indeed, the continuity equation turns into

$$\sum_i q_t^i \left(\left\langle \nabla \log q_t^i, v_f - \frac{\partial f_t(x^i)}{\partial t} \right\rangle + \nabla \cdot (v_f) \right) = 0. \tag{48}$$

From the solution and the continuity equation we write $\sum_i q_t^i \nabla \cdot (v_f)$ in two different ways.

$$\sum_i q_t^i \nabla \cdot (v_f) = -\frac{1}{\sum_i q_t^i} \left\langle \sum_i \nabla q_t^i, \sum_i q_t^i \frac{\partial f_t(x^i)}{\partial t} \right\rangle + \sum_i \left\langle \nabla q_t^i, \frac{\partial f_t(x^i)}{\partial t} \right\rangle \tag{49}$$

$$= -\left\langle \sum_i \nabla q_t^i, v_f \right\rangle + \sum_i \left\langle \nabla q_t^i, \frac{\partial f_t(x^i)}{\partial t} \right\rangle \tag{50}$$

Thus, we see that (47) is indeed a solution.

Finally, unifying v_σ and v_f, we have the full solution

$$v = -\left(\frac{\partial}{\partial t} \log \sigma_t \right) \frac{\sigma_t^2}{\sum_i q_t^i} \sum_i \nabla q_t^i + \frac{1}{\sum_i q_t^i} \sum_i q_t^i \frac{\partial f_t(x^i)}{\partial t}, \quad q_t^i(x) = \mathcal{N}(x \mid f_t(x^i), \sigma_t^2), \tag{51}$$

$$v = \frac{1}{\sum_i q_t^i} \sum_i q_t^i \left[(x - f_t(x^i)) \frac{\partial}{\partial t} \log \sigma_t + \frac{\partial f_t(x^i)}{\partial t} \right]. \tag{52}$$

Dataset	Task	BPD↓		FID↓	
		SM	AM	SM	AM
CelebA	Diffusion	2.56	3.78	4.60	18.07
CelebA	Superres	–	–	1.22	4.92
CelebA	Inpainting	–	–	2.02	10.71
CelebA	Torus	–	3.90	–	18.09
CIFAR-10	Diffusion	3.19	4.31	12.05	53.86
CIFAR-10	Superres	–	–	5.94	26.42
CIFAR-10	Colorization	–	–	5.35	7.91
CIFAR-10	Torus	–	6.42	–	39.42

Table 2: Experimental results for Action Matching (AM) and Score Matching (SM) on computer vision tasks. Diffusion and Torus map images to known distributions; hence, for them, we report negative log-likelihood in bits per dimension (BPD). For all tasks, we report FID evaluated between generated images and the test data. For CelebA, we use 20k images. For CIFAR-10, we use 10k images.

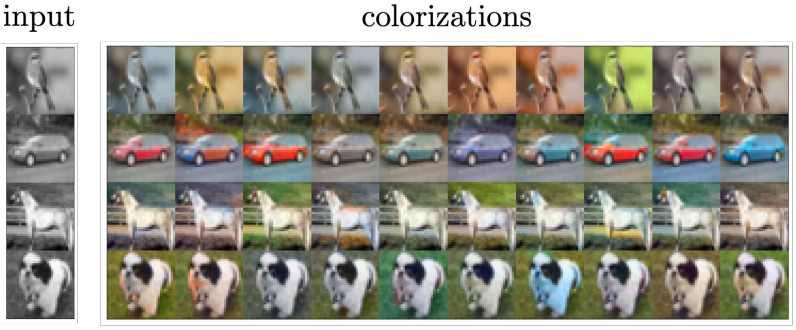

Figure 3: Illustration that Action Matching can learn one to many relations using low variance noise added to the image. Here, we sample different colorizations starting from the same grayscale input adding different samples of noise.

D Implementation Details

D.1 Details of Action Matching Generative Models

For the architecture of the neural network parameterizing s_t, we follow (Salimans & Ho, 2021). In more details, we parameterize $s_t(x)$ as $\|\text{unet}(t,x) - x\|^2$, where $\text{unet}(t,x)$ is the U-net architecture (Ronneberger et al., 2015). For the U-net architecture, we follow (Song et al., 2020b) with the only difference is that we set the channel multiplier parameter to 64 instead of 128, thus, narrowing down the architecture. We have to narrow down the architecture since Action Matching requires taking the derivative w.r.t. the inputs at each iteration, which is a downside compared to Denoising Score Matching. Otherwise the training of one model takes a week on 4 gpus. We consider the same U-net architecture for the baseline to parameterize $\nabla \log q_t$.

For diffusion, we take VP-SDE from (Song et al., 2020b), which corresponds to $\alpha_t = \exp(-\frac{1}{2}\int \beta(s)ds)$ and $\sigma_t = \sqrt{1-\exp(-\int \beta(s)ds)}$, where $\beta(s) = 0.1 + 19.9t$. For other tasks we take $\sigma_t = t$ and $\alpha_t = 1 - t$. All images are normalized to the interval $[-1, 1]$. For image generation on the torus, we first normalize the data such that every pixel is in $[0.25, 0.75]$. Thus we make sure that the shortest distance between the lowest and the largest pixel values is maximal on the circle $[0, 1]$.

Although Action Matching learns deterministic mappings, it is possible to learn one-to-many mappings by adding small amount of noise to the data. For example, each row of Fig. 3 shows that Action Matching has learned to generate different colorizations from a single grayscale CIFAR-10 image, using different noise samples added to the grayscale image in

$$x_t = \alpha_t x_0 + \sigma_t(10^{-1}\varepsilon + \text{gray}(x_0))\,. \tag{53}$$

D.2 Details on the Schrödinger Equation Simulation

For the initial state of the dynamics

$$i\frac{\partial}{\partial t}\psi(x,t) = -\frac{1}{\|x\|}\psi(x,t) - \frac{1}{2}\nabla^2\psi(x,t)\,, \tag{54}$$

we take the following wavefunction

$$\psi(x, t=0) \propto \psi_{32-1}(x) + \psi_{210}(x), \quad \text{and} \quad q^*_{t=0}(x) = |\psi(x, t=0)|^2, \tag{55}$$

where n, l, m are quantum numbers and ψ_{nlm} is the eigenstate of the corresponding Hamiltonian (see Griffiths & Schroeter (2018)). For all the details on sampling and the exact formulas for the initial state, we refer the reader to the code github.com/action-matching. We evolve the initial state for $T = 14 \cdot 10^3$ time units in the system $\hbar = 1, m_e = 1, e = 1, \varepsilon_0 = 1$ collecting the dataset of samples from q^*_t. For the time discretization, we take 10^3 steps; hence, we sample every 14 time units.

To evaluate each method, we collect all the generated samples from the distributions $q_t,\ t \in [0, T]$ comparing them with the samples from the training data. For the distance metric, we measure the MMD distance (Gretton et al., 2012) between the generated samples and the training data at 10 different timesteps $t = \frac{k}{10}T,\ k = 1, \ldots, 10$ and average the distance over the timesteps. For the Annealed Langevin Dynamics, we set the number of intermediate steps for $M = 5$, and select the step size dt by minimizing MMD using the exact scores $\nabla \log q_t(x)$.

For all methods, we use the same architecture, which is a multilayer perceptron with 5 layers 256 hidden units each. The architecture $h(t, x)$ takes $x \in \mathbb{R}^3$ and $t \in \mathbb{R}$ and outputs 3-d vector, i.e. $h(t, x) : \mathbb{R} \times \mathbb{R}^3 \to \mathbb{R}^3$. For the score-based models it already defines the score, while for action matching we use $s_t(x) = \|h(t, x) - x\|^2$ as the model and the vector field is defined as $\nabla s_t(x)$.

Algorithm 2 Annealed Langevin Dynamics

Require: score model $s_t(x)$, step size dt, number of intermediate steps M
Require: initial samples $x^i_0 \in \mathbb{R}^d$
  for time steps $t \in (0, T]$ **do**
    set the target distribution q_t, such that $s_t(x) \approx \nabla \log q_t(x)$
    for intermediate steps $j \in 1, \ldots, M$ **do**
      $\varepsilon^i \sim \mathcal{N}(0, \mathbf{1})$
      $x^i_t = x^i_t + \frac{dt}{2} s_t(x^i_t) + \sqrt{dt} \cdot \varepsilon^i$
    end for
    save samples x^i_t
  end for
  return samples $\{x^i_t\}_{t=0}^T$

E Samples of Action Matching Generative Models

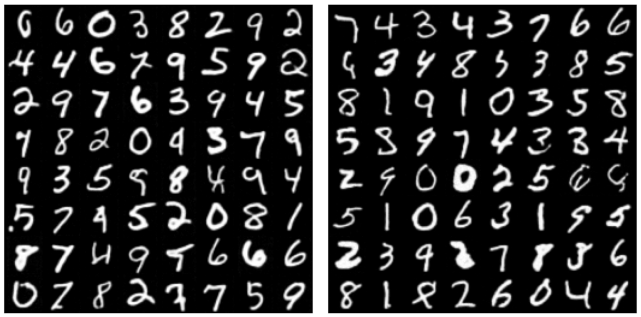

Figure 4: Action Matching generated images on MNIST for diffusion (on the left), for torus (on the right). Prior distribution is not shown.

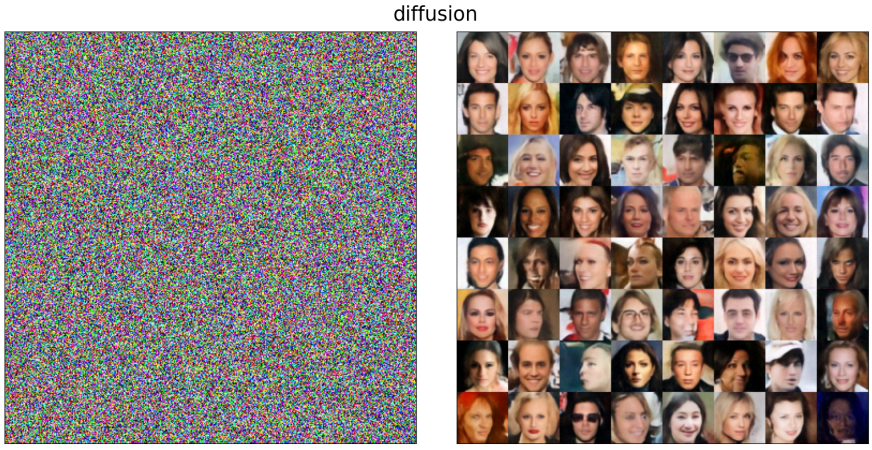

Figure 5: Action Matching on CelebA for diffusion. Prior distribution is on the left, generated images are on the right.

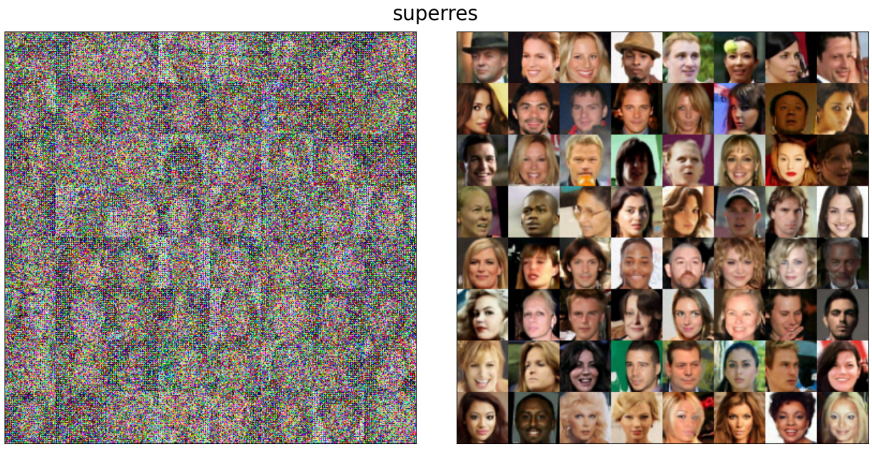

Figure 6: Action Matching on CelebA for superres. Prior distribution is on the left, generated images are on the right.

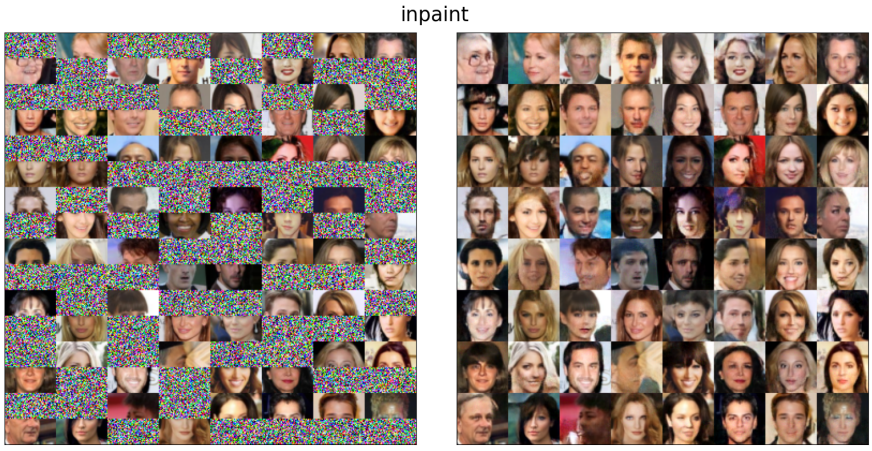

Figure 7: Action Matching on CelebA for inpaint. Prior distribution is on the left, generated images are on the right.

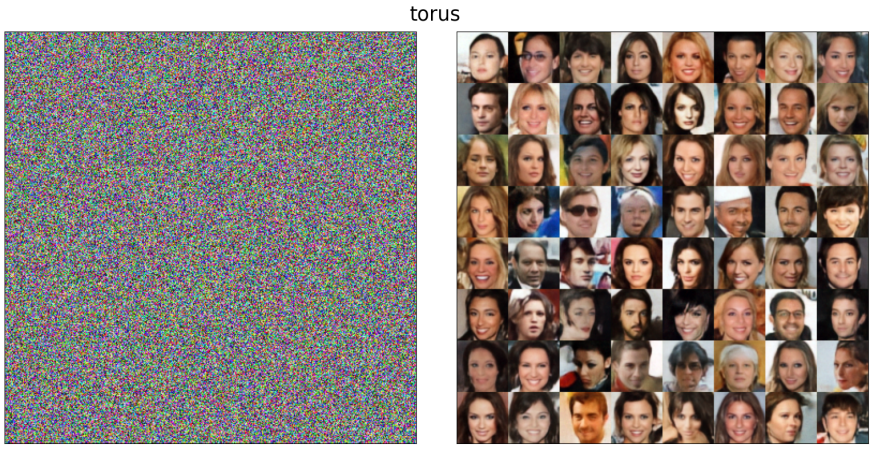

Figure 8: Action Matching on CelebA for torus. Prior distribution is on the left, generated images are on the right.

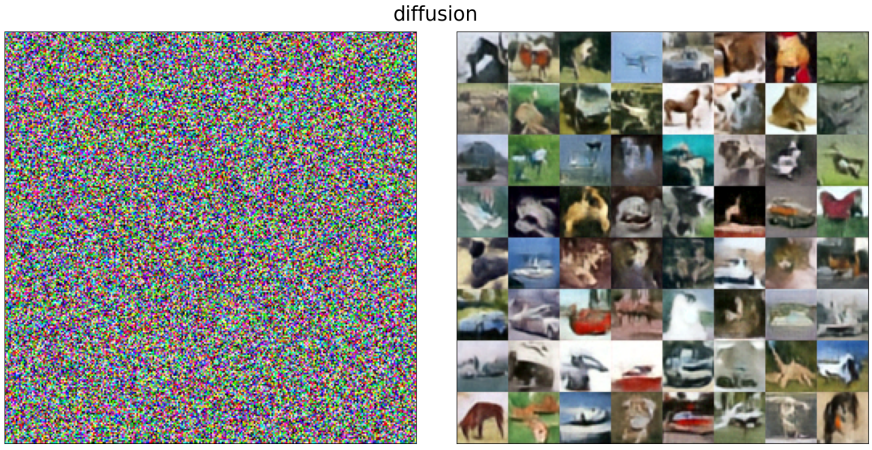

Figure 9: Action Matching on CIFAR-10 for diffusion. Prior distribution is on the left, generated images are on the right.

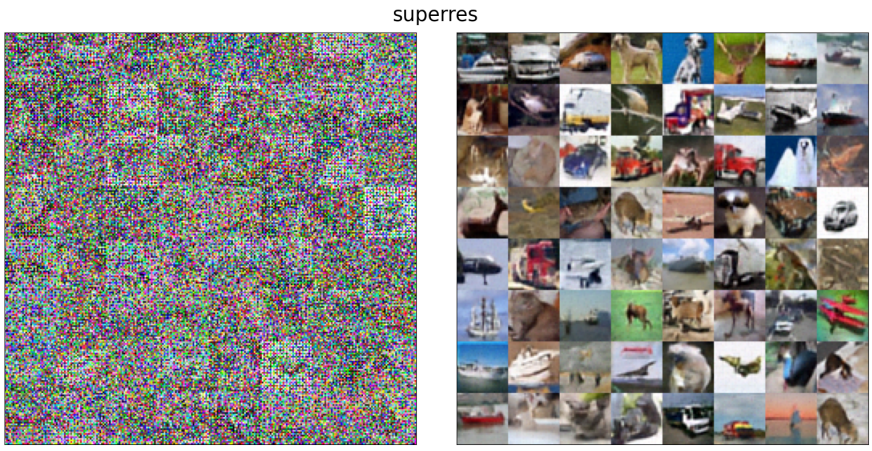

Figure 10: Action Matching on CIFAR-10 for superres. Prior distribution is on the left, generated images are on the right.

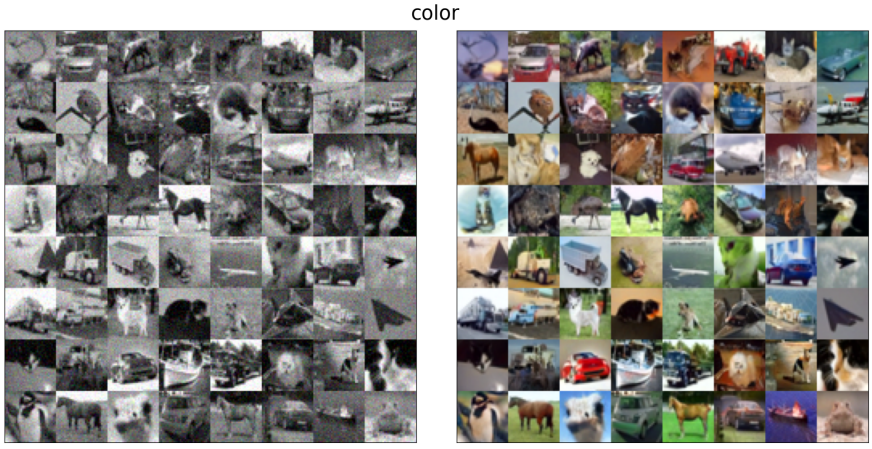

Figure 11: Action Matching on CIFAR-10 for colorization. Prior distribution is on the left, generated images are on the right.

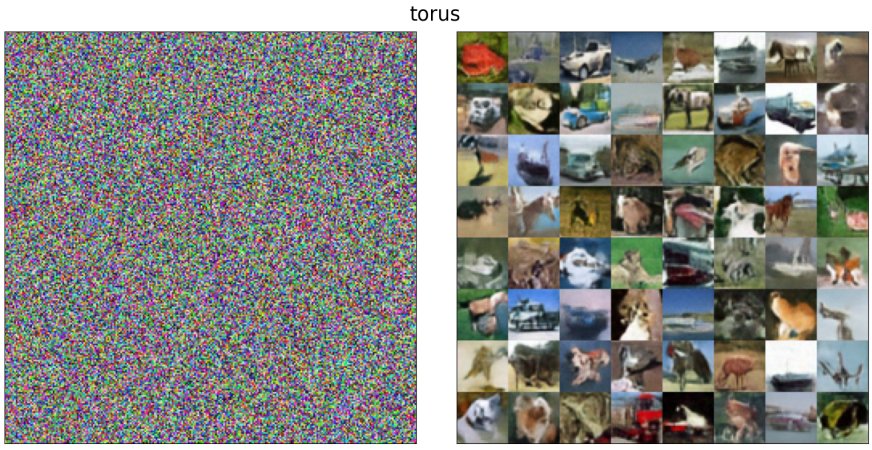

Figure 12: Action Matching on CIFAR-10 for torus. Prior distribution is on the left, generated images are on the right.