Neural Diffusion Models

Grigory Bartosh [1] **Dmitry Vetrov** [2] **Christian A. Naesseth** [1]

Abstract

Diffusion models have shown remarkable performance on many generative tasks. Despite recent success, most diffusion models are restricted in that they only allow linear transformation of the data distribution. In contrast, broader family of transformations can help train generative distributions more efficiently, simplifying the reverse process and closing the gap between the true negative log-likelihood and the variational approximation. In this paper, we present Neural Diffusion Models (NDMs), a generalization of conventional diffusion models that enables defining and learning time-dependent non-linear transformations of data. We show how to optimise NDMs using a variational bound in a simulation-free setting. Moreover, we derive a time-continuous formulation of NDMs, which allows fast and reliable inference using off-the-shelf numerical ODE and SDE solvers. Finally, we demonstrate the utility of NDMs through experiments on many image generation benchmarks, including MNIST, CIFAR-10, downsampled versions of ImageNet and CelebA-HQ. NDMs outperform conventional diffusion models in terms of likelihood, achieving state-of-the-art results on ImageNet and CelebA-HQ, and produces high-quality samples.

1. Introduction

Generative models are a powerful class of probabilistic machine learning models with a wide range of applications from e.g. art and music to medicine and physics (Tomczak, 2022; Creswell et al., 2018; Papamakarios et al., 2021; Yang et al., 2022). Generative models learn to mimic the underlying probability distribution of a given data set and can generate novel samples that are similar to the original data. They can for example be used for data augmentation, as well as for unsupervised learning.

Diffusion models have emerged as a family of generative models that excel at several generative tasks (Sohl-Dickstein et al., 2015; Ho et al., 2020). They parameterize the data model through an iterative refinement process, the *reverse process*, that builds up the data step-by-step from pure noise. For training purposes an auxiliary noising process, the *forward process*, is introduced that successively adds noise to data. The reverse process is then optimized to resemble the forward process. Despite success in various domains (Sohl-Dickstein et al., 2015; Ho et al., 2020; Saharia et al., 2021; Popov et al., 2021; Watson et al., 2022; Trippe et al., 2023), a key limitation of most existing diffusion models is that they rely on a fixed and pre-specified forward process that is unable to adapt to the specific task or data at hand. At the same time there are many works (Hoogeboom & Salimans, 2023; Rombach et al., 2022; Lipman et al., 2023) that improve performance of diffusion models by modifications of the forward processes.

In this paper we develop Neural Diffusion Models (NDMs), a general framework that enables non-linear, time-dependent and learnable data transformations. We extend the approach by Song et al. (2021a) and construct the forward process as a non-Markovian sequence of latent variables; each latent variable is constructed through a transformation of the data to which we then inject noise. This is then leveraged in the corresponding reverse process. To train NDMs efficiently we generalize the diffusion objective while keeping it a simulation-free bound on the log-likelihood. Furthermore, we derive the time-continuous analogue of the objective function as well as the stochastic differential equation (SDE) and ordinary differential equation (ODE) corresponding to the reverse process.

We demonstrate how NDMs generalizes several existing diffusion models and then propose a new model with learnable transformations of data parameterized by a neural network. To illustrate the empirical properties of NDMs we provide experimental results on a synthetic data as well as on MNIST, CIFAR-10, downsampled ImageNet and CelebA-HQ image datasets. NDMs consistently outperforms baselines in terms of negative log-likelihood, achieving state-of-the-art results for diffusion models on ImageNet 32 and 64, as well as CelebA-HQ.

[1]University of Amsterdam [2]Constructor University, Bremen. Correspondence to: Grigory Bartosh <g.bartosh@uva.nl>, Dmitry Vetrov <dvetrov@constructor.university>, Christian A. Naesseth <c.a.naesseth@uva.nl>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

The main motivation for NDMs is improved likelihood and density estimation, crucial for applications to compression (MacKay, 2003), semi-supervised learning (Dai et al., 2017), adversarial purification (Song et al., 2017), and many others. However, for completeness we also study the impact of NDMs on image generation quality. We find that for small to medium number of steps NDMs achieves better image generation quality than denoising diffusion probabilistic models (DDPMs) (Ho et al., 2020), and comparable results for a large number of steps.

Finally, we demonstrate that NDMs allows learning simpler generative dynamics like dynamical optimal transport, which conventional diffusion models are incapable of learning.

We summarize the contributions as follows:

1. We propose neural diffusion models or NDMs, a new framework that generalizes conventional diffusion models in both discrete and continuous time settings.
2. We develop an objective function to optimize NDMs that upper bounds the negative log-likelihood and study its properties.
3. We demonstrate the utility of NDMs with learnable transformations in terms of consistently and significantly improved log-likelihood, as well as better or comparable generation quality.

2. Background

Diffusion models are generative models that make use of latent variables. Given a sample from the data distribution $\mathbf{x} \sim q(\mathbf{x})$, we define a forward noising process that produces latent variables $\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_T$. In contrast, the reverse generative process reverts the forward process, starting by first generating the same latent variables and then data $\mathbf{x}$.

The standard approach to specify the forward process is as a linear Gaussian Markov chain (Sohl-Dickstein et al., 2015; Ho et al., 2020). However, we can also use an implicit definition of the forward process from Song et al. (2021a). This will turn out to be useful for our purposes and is what we focus on here. To construct the implicit forward process we first define the marginal distributions $q(\mathbf{z}_t|\mathbf{x})$. Using these marginal distributions we can define the joint distribution of all latent variables $\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_T$ as follows:

$$q(\mathbf{z}_{0:T}|\mathbf{x}) = q(\mathbf{z}_T|\mathbf{x}) \prod_{t=1}^{T} q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x}),$$

$$\text{with} \quad q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x}) \quad \text{such that}$$

$$q(\mathbf{z}_{t-1}|\mathbf{x}) = \int q(\mathbf{z}_t|\mathbf{x}) q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x}) d\mathbf{z}_t. \tag{1}$$

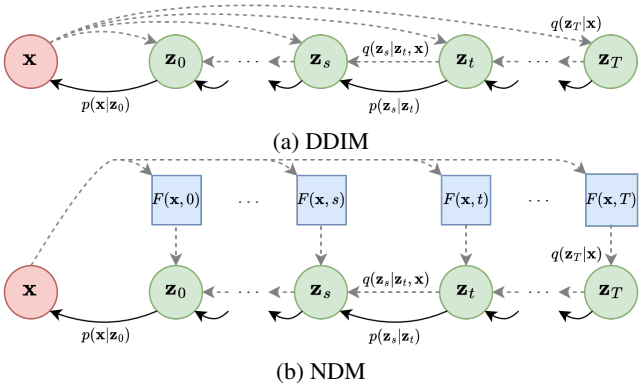


Figure 1. The directed graphical models of DDIM and NDM.

Here we make use of the posterior distribution $q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})$ instead of the regular forward distribution $q(\mathbf{z}_t|\mathbf{z}_{t-1})$. Due to the dependence also on the data $\mathbf{x}$ it is a non-Markovian forward process (see Figure 1a). In general the forward process is considered fixed and has no trainable parameters. Moreover, it is specified in such a way that $q(\mathbf{z}_0|\mathbf{x}) \approx \delta(\mathbf{z}_0 - \mathbf{x})$ and $q(\mathbf{z}_T|\mathbf{x}) \approx \mathcal{N}(\mathbf{z}_T; 0, I)$. So if $q(\mathbf{z}_{t-1}|\mathbf{z}_t)$ was available we could sample $\mathbf{z}_T \sim \mathcal{N}(\mathbf{z}_T; 0, I)$ and run the reverse process to get $\mathbf{z}_0 \sim q(\mathbf{z}_0) \approx q(\mathbf{x})$. However, the distribution $q(\mathbf{z}_{t-1}|\mathbf{z}_t)$ depends implicitly on the data distribution $q(\mathbf{x})$ and thus has a complicated form, so we instead approximate the reverse process using a Markov chain with distribution $p_\theta(\mathbf{z}_{0:T})$:

$$p_\theta(\mathbf{z}_{0:T}) = p(\mathbf{z}_T) \prod_{t=1}^{T} p_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t), \tag{2}$$

where $p(\mathbf{z}_T) = \mathcal{N}(\mathbf{z}_T; 0, I)$.

The combination of the forward process q and the reverse process p_θ is a form of (hierarchical) variational autoencoder (Kingma & Welling, 2014; Rezende et al., 2014). Therefore, it can be trained by optimizing the usual variational bound on the negative log-likelihood. In the case of diffusion models, it can be written as follows (see Section A of Ho et al. (2020)):

$$\mathbb{E}_q \Bigg[\underbrace{\mathrm{D}_{\mathrm{KL}}\Big(q(\mathbf{z}_T|\mathbf{x}) || p(\mathbf{z}_T)\Big)}_{\mathcal{L}_{\text{prior}}} - \underbrace{\log p_\theta(\mathbf{x}|\mathbf{z}_0)}_{\mathcal{L}_{\text{rec}}} + \underbrace{\sum_{t=1}^{T} \mathrm{D}_{\mathrm{KL}}\Big(q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x}) || p_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t)\Big)}_{\mathcal{L}_{\text{diff}}} \Bigg] \tag{3}$$

Since the process q and the distribution $p_\theta(\mathbf{z}_T) = p(\mathbf{z}_T)$ are fixed, the prior term $\mathcal{L}_{\text{prior}}$ does not depend on the parameters θ, so it can be omitted. The distribution $p_\theta(\mathbf{x}|\mathbf{z}_0)$ is often take to be a Gaussian distribution, with low variance,

Table 1. Summary of existing diffusion models as instances of Neural Diffusion Models (NDM). See extended table in Appendix B.

Model	Distribution $q(\mathbf{z}_t\|x)$	NDM's $F(\mathbf{x}, t)$	Comment
DDPM (Ho et al., 2020) / DDIM (Song et al., 2021a)	$\mathcal{N}\Big(\mathbf{z}_t; \alpha_t \mathbf{x}, \sigma_t^2 I\Big)$	$\mathbf{x}$	
Flow Matching OT (Lipman et al., 2023)	$\mathcal{N}\Big(\mathbf{z}_t; \alpha_t \mathbf{x}, \sigma_t^2 I\Big)$	$\mathbf{x}$	$\alpha_t = t$, $\sigma_t = 1 - (1 - \sigma_{\min})t$
VDM (Kingma et al., 2021)	$\mathcal{N}\Big(\mathbf{z}_t; \alpha_t \mathbf{x}, \sigma_t^2 I\Big)$	$\mathbf{x}$	$\alpha_t^2 = \mathrm{sigmoid}(-\gamma_\eta(t))$, $\sigma_t^2 = \mathrm{sigmoid}(\gamma_\eta(t))$
Soft Diffusion (Daras et al., 2022)	$\mathcal{N}\Big(\mathbf{z}_t; C_t \mathbf{x}, s_t^2 I\Big)$	$C_t \mathbf{x}$	$\alpha_t = 1$, $\sigma_t^2 = s_t^2$
LSGM (Vahdat et al., 2021)	$\mathcal{N}\Big(\mathbf{z}_t; \alpha_t E(\mathbf{x}), \sigma_t^2 I\Big)$	$E(\mathbf{x})$	$p(x\|z_0) = \mathcal{N}\Big(x; aD(z_0), \sigma^2\Big)$

for continuous data and a dequantization distribution for discrete data. Thus, also the reconstruction term $\mathcal{L}_{\text{rec}}$ does not depend on the parameter θ.

This means that the only part that depends on the model parameters θ is the diffusion term $\mathcal{L}_{\text{diff}}$. It is a sum of Kullback–Leibler (KL) divergences between posterior distributions in the forward process $q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})$ and the distributions $p_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t)$ from the reverse process. In the general case this KL divergence is intractable, so the standard choice here is to set the marginal conditional distributions to be Gaussian, i.e. $q(\mathbf{z}_t|\mathbf{x}) = \mathcal{N}(\mathbf{z}_t; \alpha_t \mathbf{x}, \sigma_t^2 I)$. The posterior distribution then takes the form:

$$q(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x}) = \mathcal{N}\left(\mathbf{z}_s; \mu_{s|t}, \tilde{\sigma}_{s|t}^2 I\right), \quad \text{for} \quad 0 \le s \le t \le T,$$

$$\mu_{s|t} = \alpha_s \mathbf{x} + \frac{\sqrt{\sigma_s^2 - \tilde{\sigma}_{s|t}^2}}{\sigma_t}(\mathbf{z}_t - \alpha_t \mathbf{x}). \tag{4}$$

Note that here we allow for an arbitrary choice of time grid, i.e. s and t, whereas above it was equidistant. It is straightforward to check that such a posterior distribution satisfies (1) for any $\tilde{\sigma}_{s|t}^2 \le \sigma_s^2$. The exact schedule of $\tilde{\sigma}_{s|t}^2$ is a user design choice.

Finally, the reverse distribution is set to $p_\theta(\mathbf{z}_s|\mathbf{z}_t) = q(\mathbf{z}_s|\mathbf{z}_t, \hat{\mathbf{x}}_\theta(\mathbf{z}_t, t))$, where $\hat{\mathbf{x}}_\theta(\mathbf{z}_t, t)$ is the model's prediction of $\mathbf{x}$. Since $q(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x})$ and $p_\theta(\mathbf{z}_s|\mathbf{z}_t)$ are both Gaussian distributions, we can compute the KL divergences in $\mathcal{L}_{\text{diff}}$ in closed form.

This choice of forward and reverse processes, resulting in analytic expressions for the diffusion terms given data, is what makes diffusion models a *simulation-free* approach. Simulation-free means that we do not have to sample all latent variables for each optimization step. Rather than calculating all individual terms in $\mathcal{L}_{\text{diff}}$, we can uniformly sample t and optimize only a subset of KL divergences using stochastic gradient descent.

By choosing a specific value for $\tilde{\sigma}_{s|t}^2$, we can obtain equality between the processes of DDPM and DDIM (see section 4.1 of Song et al. (2021a)). Furthermore, as Song et al. (2021c) demonstrated, when the number of steps T in DDPM goes to infinity, we can transition to continuous time. In this scenario, the reverse process can be described using a Stochastic Differential Equation (SDE):

$$\begin{aligned} d\mathbf{z}_t &= [r(t)\mathbf{z}_t - g^2(t) s_\theta(\mathbf{z}_t, t)]dt + g(t) d\mathbf{w}_t, \\ s_\theta(\mathbf{z}_t, t) &= \frac{\alpha_t \hat{\mathbf{x}}_\theta(\mathbf{z}_t, t) - \mathbf{z}_t}{\sigma_t^2}, \quad r(t) = \frac{d \log \alpha_t}{dt}, \\ g^2(t) &= \frac{d\sigma_t^2}{dt} - 2\frac{d \log \alpha_t}{dt}\sigma_t^2, \end{aligned} \tag{5}$$

with time running backwards from $t = 1$ to $t = 0$. This formulation allows us to switch to the equivalent ODE and to use different SDE and ODE solvers for sampling and density estimation.

3. Neural diffusion models

In theory, we can view diffusion models as a special type of hierarchical variational autoencoders (VAE). From this perspective, the conventional diffusion model resembles a VAE with a fixed variational distribution, in which the latent variables are inferred using scaling of data points and injecting of Gaussian noise. Such a formulation limits diffusion models in terms of the flexibility of the latent space. Introducing a more flexible (and learnable) distribution of latent variables could effectively reduce the gap between the log-likelihood and the variational bound. In practical

Algorithm 1 Learning NDM

Require: $q(\mathbf{x})$, F_φ, $\hat{\mathbf{x}}_\theta$
  for learning iterations **do**
    $\mathbf{x} \sim q(\mathbf{x})$, $t \sim U[1, T]$, $\varepsilon \sim \mathcal{N}(0, I)$
    $\mathbf{z}_t \sim q_\varphi(\mathbf{z}_t|\mathbf{x})$
    $\mathcal{L} = \mathcal{L}_{\text{rec}} + \mathcal{L}_{\text{diff}} + \mathcal{L}_{\text{prior}}$
    Gradient step on θ and φ w.r.t. $\mathcal{L}$
  end for

Algorithm 2 Sampling from NDM

Require: F_φ, $\hat{\mathbf{x}}_\theta$
  $\mathbf{z}_T \sim \mathcal{N}(0, I)$
  for $t = T, \ldots, 1$ **do**
    $\hat{\mathbf{x}} = \hat{\mathbf{x}}_\theta(\mathbf{z}_t, t)$
    $\mathbf{z}_{t-1} \sim q_\varphi(\mathbf{z}_{t-1}|\mathbf{z}_t, \hat{\mathbf{x}})$
  end for
  $x \sim p(\mathbf{x}|\mathbf{z}_0)$

terms, a more flexible forward process might simplify the task of learning the reverse (generative) process, thereby enhancing model quality. To overcome this limitation of conventional diffusion models, we propose a general form of transformations of data that allows to define and learn distributions on the latent space.

In this section, we introduce the Neural Diffusion Models (NDMs) – a simulation-free framework that generalises conventional diffusion models. The key idea in NDMs is to apply a time-dependent transformation $F_\varphi(\mathbf{x}, t)$ to the data $\mathbf{x}$ at each step of forward process before injecting noise. Previous diffusion models arise as special cases when the data transformation is either linear, time-independent, or pre-defined non-linear (see Table 1). In contrast, the NDM can work with any time-dependent transformation of data and may be learned end-to-end. In Section 4 we provide experimental results with $F_\varphi(\mathbf{x}, t)$ parameterized by neural networks.

3.1. Model definition and variational objective

We introduce NDMs constructively. First, we define the desired marginal distributions:

$$q_\varphi(\mathbf{z}_t|\mathbf{x}) = \mathcal{N}\Big(\mathbf{z}_t; \alpha_t F_\varphi(\mathbf{x}, t), \sigma_t^2 I\Big), \tag{6}$$

where $F_\varphi(\mathbf{x}, t) : \mathbb{R}^d \times [0, T] \mapsto \mathbb{R}^d$ is a function parameterized by φ that applies a time-dependent transform to the data point $\mathbf{x}$. We adapt the approach from DDIM, as described in Section 2, and choose the following posterior distribution that satisfies (6) (we provide derivation and proof in Appendix A.1):

$$q_\varphi(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x}) = \mathcal{N}\left(\mathbf{z}_s; \mu^{F_\varphi}_{s|t}, \tilde{\sigma}^2_{s|t} I\right), \tag{7}$$

$$\mu^{F_\varphi}_{s|t} = \alpha_s F_\varphi(\mathbf{x}, s) + \frac{\sqrt{\sigma_s^2 - \tilde{\sigma}^2_{s|t}}}{\sigma_t}\Big(\mathbf{z}_t - \alpha_t F_\varphi(\mathbf{x}, t)\Big),$$

for $0 \leq s \leq t \leq T$ where $\tilde{\sigma}^2_{s|t} \leq \sigma^2_s$ is a design choice. Using this posterior we can define an implicit forward process according to (1) (see Figure 1b). This forward process provides access to both marginal and posterior distributions just like in the DDIM framework (Song et al., 2021a). The corresponding NDM variational objective has the following form:

$$\mathbb{E}_{q_\varphi}\Bigg[\underbrace{\mathrm{D}_{\mathrm{KL}}\Big(q_\varphi(\mathbf{z}_T|\mathbf{x})||p(\mathbf{z}_T)\Big)}_{\mathcal{L}_{\text{prior}}} - \underbrace{\log p_\theta(\mathbf{x}|\mathbf{z}_0)}_{\mathcal{L}_{\text{rec}}} + \underbrace{\sum_{t=1}^{T} \mathrm{D}_{\mathrm{KL}}\Big(q_\varphi(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})||p_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t)\Big)}_{\mathcal{L}_{\text{diff}}}\Bigg]. \tag{8}$$

While the objective has the same form as in DDIM (3), the individual terms are different. If the transformation $F_\varphi(\mathbf{x}, t)$ is actually parameterized by learnable parameters φ, the prior term $\mathcal{L}_{\text{prior}}$ and the reconstruction term $\mathcal{L}_{\text{rec}}$ depend on the parameter φ as well. Therefore, in that case these terms cannot be excluded from the optimization process.

For the standard parameterization of the reverse process through approximate posteriors $p_\theta(\mathbf{z}_s|\mathbf{z}_t) = q_\varphi(\mathbf{z}_s|\mathbf{z}_t, \hat{\mathbf{x}}_\theta(\mathbf{z}_t, t))$ the KL divergences in the diffusion term $\mathcal{L}_{\text{diff}}$ are (see Appendix A.2):

$$\mathrm{D}_{\mathrm{KL}}\Big(q_\varphi(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x})||p_\theta(\mathbf{z}_s|\mathbf{z}_t)\Big) = \frac{1}{2\tilde{\sigma}^2_{s|t}}\left\|\alpha_s\Big(F_\varphi(\mathbf{x}, s) - F_\varphi(\hat{\mathbf{x}}_\theta(\mathbf{z}_t, t), s)\Big) + \frac{\sqrt{\sigma_s^2 - \tilde{\sigma}^2_{s|t}}}{\sigma_t}\alpha_t\Big(F_\varphi(\hat{\mathbf{x}}_\theta(\mathbf{z}_t, t), t) - F_\varphi(\mathbf{x}, t)\Big)\right\|_2^2. \tag{9}$$

Note a distinction between the objectives of NDM and DDIM here. In the case of DDIM, the model tries to accurately predict the data point $\mathbf{x}$. In contrast, NDM aims to predict the *transformed* data point $F_\varphi(\mathbf{x}, t)$. Despite this change, NDM's optimization remains simulation-free, so we can efficiently train the NDM by sampling time steps and calculating corresponding KL divergences. We summarise the training and sampling procedures in Algorithms 1 and 2.

Given that NDM is a generalization of DDIM, we can leverage the same techniques for inference. Specifically, we can adjust the number of intermediate time steps, the schedule

of $\tilde{\sigma}^2_{s|t}$ as well as sampling with various dynamics, including a deterministic dynamic corresponding to $\tilde{\sigma}^2_{s|t} = 0$.

3.2. Continuous time NDMs

We previously formulated NDMs in the discrete time setting with T steps. However, like conventional diffusion models, we can let the number of steps T go to infinity and switch to continuous time. In this case, the set of time steps $\{0, 1, \ldots, T\}$ transforms to the range $[0, 1]$ and the diffusion term of the objective reduces to an expectation over time (see derivation in Appendix A.4):

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{q(\mathbf{x})u(t)q(\mathbf{z}_t|\mathbf{x})}\Bigg[\frac{1}{g^2(t)}\Bigg\|\alpha_t\Big(\dot{F}_\varphi(\mathbf{x}, t) - \dot{F}_\varphi\big(\hat{\mathbf{x}}_\theta(\mathbf{z}_t, t), t\big)\Big) + \frac{1}{2}\left(\frac{\partial \sigma_t^2}{\partial t} - 2r(t)\sigma_t^2 + g^2(t)\right) \cdot \Big(s(\mathbf{x}, \mathbf{z}_t, t) - s\big(\hat{\mathbf{x}}_\theta(\mathbf{z}_t, t), \mathbf{z}_t, t\big)\Big)\Bigg\|_2^2\Bigg], \quad (10)$$

where $r(t) = \frac{\partial \log \alpha_t}{\partial t}$, $g^2(t) = \dot{\nu}_t \sigma_t^2$, and $s(\mathbf{x}, \mathbf{z}_t, t) = \frac{\alpha_t F_\varphi(\mathbf{x}, t) - \mathbf{z}_t}{\sigma_t^2}$.

Similar to training a discrete time NDM, we can train a continuous time NDMs by sampling time. In our experiments we use importance sampling (Song et al., 2021b) and sample time from a distribution proportional to $\frac{1}{g^2(t)}$.

Note, that we may not have access to the partial derivative of the transformation $F_\varphi(\cdot, t)$ with respect to t in closed form. However, for any differentiable $F_\varphi(\cdot, t)$ we can use Jacobian-Vector product (Smale & Hirsch, 1974) to obtain this derivative.

The discrete time reverse process also becomes a continuous time process, described by a Stochastic Differential Equation (SDE). If we parameterize the noise injection in the posterior distribution as $\tilde{\sigma}^2_{s|t} = \sigma_s^2(1 - e^{\nu_s - \nu_t})$, we obtain the following SDE (see derivation in Appendix A.3):

$$d\mathbf{z}_t = \Big[\alpha_t \dot{F}_\varphi(\hat{\mathbf{x}}_\theta(\mathbf{z}_t, t), t) + r(t)\mathbf{z}_t - \frac{1}{2}\Big(g^2(t) - 2r(t)\sigma_t^2\Big) s_\theta(\mathbf{z}_t, t)\Big] dt + g(t) d\mathbf{w}, \quad (11)$$

where $s_\theta(\mathbf{z}_t, t) = \frac{\alpha_t F_\varphi(\hat{\mathbf{x}}_\theta(\mathbf{z}_t, t), t) - \mathbf{z}_t}{\sigma_t^2}$.

By changing the function ν_t, we can obtain different dynamics. In the extreme case where ν_t is equal to a constant we have deterministic dynamics described by an ODE. This enables the use of SDE or ODE solvers for inference. Moreover, we can estimate densities by considering the model as a continuous normalizing flow (Chen et al., 2018) in the deterministic case.

4. Experiments

We present empirical results for the proposed Neural Diffusion Models with learnable transformations on a synthetic datasets as well as multiple image datasets. Qualitatively, NDMs learn transformations that simplify the data distribution, leading to predictions of $\mathbf{x}$ that are more aligned with the data. Quantitatively, NDMs consistently outperform the baseline in terms of likelihood, achieving state-of-the-art diffusion model results for ImageNet and CelebA-HQ. Moreover, for a small to medium number of steps, NDMs achieve better image generation quality than DDPM, while being comparable for a large number of steps. We also provide a proof of concept experiment that demonstrates that NDMs can learn simple generative trajectories, something conventional diffusion models are incapable of learning.

4.1. Implementation details

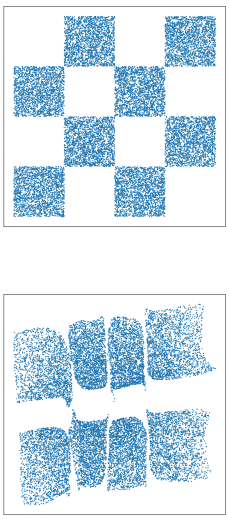
We demonstrate NDMs with learnable transformations on the MNIST (Deng, 2012), CIFAR-10 (Krizhevsky et al., 2009), downsampled ImageNet (Deng et al., 2009; Van Den Oord et al., 2016) and CelebA-HQ-256 (Karras et al., 2017) datasets. In all experiments we use same neural network architectures to parameterize both the generative process and the transformations F_φ. In experiments with images we use the U-Net architecture from Dhariwal & Nichol (2021). To ensure consistency with Song et al. (2021c;b), we apply horizontal flipping as a data augmentation technique for training models on CIFAR-10 and ImageNet. Unless otherwise stated, we utilize the DDPM variance-preserving schedule of noise injection for α_t and σ_t^2. For density estimation of discrete data we use uniform dequantization.

In the experiments we report negative log-likelihood (NLL) in bits per dimension (BPD), negative evidence lower bound (NELBO) (8), and sample quality as measured by the Frechet Inception Distance (FID) (Heusel et al., 2017). We calculate NLL by integrating the corresponding ODEs using the RK45 solver from Dormand & Prince (1980), and both NLL and NELBO are calculated on test data. For FID we report the score computed using 50k generated images.

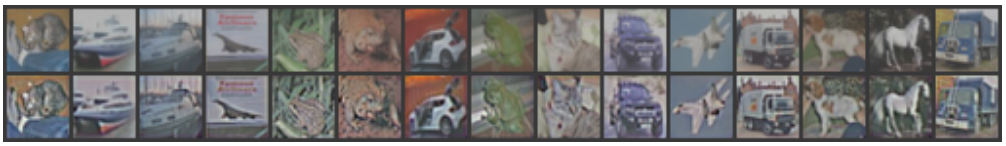
In Section 3 we parameterize the reverse process through $\hat{\mathbf{x}}_\theta$ function. However, in practice we reparameterize the generative process in terms of prediction of injected noise. For a detailed description of parameterizations and other experimental details, please refer to Appendix C.

4.2. Learned transformations

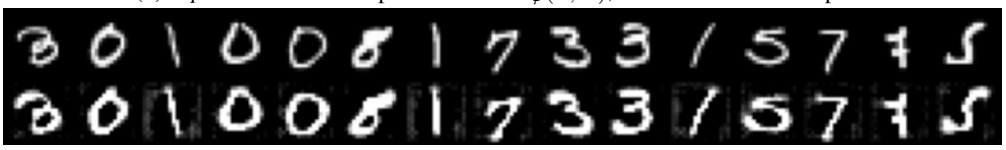
Let us examine some of the transformations that NDM learns. Figure 2a-2c illustrates the transformations that NDM learns for the 2D checkerboard distribution, MNIST, and CIFAR-10 datasets. For the checkerboard, we observe that F_φ learns to transform the interleaved pattern into a



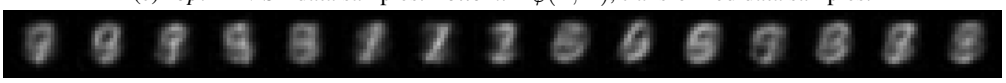
(a) *Top:* Data $\mathbf{x}$. *Bottom:* $F_\varphi(\mathbf{x}, T)$, transformed data.



(b) *Top:* CIFAR data samples. *Bottom:* $F_\varphi(\mathbf{x}, T)$, transformed data samples.



(c) *Top:* MNIST data samples. *Bottom:* $F_\varphi(\mathbf{x}, T)$, transformed data samples.



(d) DDPM predictions $\hat{\mathbf{x}}_\theta(\mathbf{z}_T, T)$ for $\mathbf{z}_T \sim \mathcal{N}(\mathbf{z}_T; 0, I)$.

(e) NDM predictions $\hat{\mathbf{x}}_\theta(\mathbf{z}_T, T)$ for $\mathbf{z}_T \sim \mathcal{N}(\mathbf{z}_T; 0, I)$.

Figure 2. Learned transforms for the 2D checkerboard distribution (*left*). Learned transforms for CIFAR-10 and MNIST (*top right*), as well as predictions for MNIST (*bottom right*). NDM learns useful forward transformations and more accurately predicts the data from injected noise.

Table 2. We report Negative Log-Likelihood (NLL) in Bits Per Dimension (BPD) on the test sets for CIFAR10, ImageNet 32x32, and ImageNet 64x64. Our results were obtained using the continuous-time formulation of our model, integrated via the corresponding Ordinary Differential Equation (ODE), as detailed in Section 3.2.

Model	**CIFAR10**	**ImageNet 32**	**ImageNet 64**
DDPM (Ho et al., 2020)	3.69		
Improved DDPM (Nichol & Dhariwal, 2021)	2.94		3.54
VDM (Kingma et al., 2021)	**2.65**	3.72	3.40
Score SDE (Song et al., 2021c)	2.99		
Score Flow (Song et al., 2021b)	2.83	3.76	
NDM (ours)	2.70	**3.55**	**3.35**

non-interleaved one. In the case of the grayscale digits of the MNIST dataset, F_φ learns to highlight the distinctive features of the numbers. It thickens the lines and even creates bubbles at the corners. For the color images of CIFAR-10, F_φ learns to increase the image contrast. In all cases, our model learns a way to simplify the data distribution. These transformations may enable the reverse process to transition more smoothly from simple distributions to complex ones.

Furthermore, we would like to emphasize the difference between the predictions of $\mathbf{x}$ that NDM and DDPM makes. Figure 2d and Figure 2e shows the predictions $\hat{\mathbf{x}}_\theta(\mathbf{z}_T, T)$ generated by NDM and DDPM models trained on the MNIST dataset. In each case, the model samples from a standard normal distribution $\mathbf{z}_T \sim \mathcal{N}(\mathbf{z}_T; 0, I)$ and based on this value tries to predict $\mathbf{x}$. Therefore, we do not expect these samples to be of high quality. However, as we can see, NDM's predictions of $\mathbf{x}$ are much more similar to the data distribution than DDPM's predictions. We attribute this behavior to the fact that our model aims to predict not the datapoint $\mathbf{x}$, but the transformed datapoint $F_\varphi(\mathbf{x}, t)$. Thus, to make better predictions of the transformed datapoint, it may be critical to generate predictions of $\mathbf{x}$ that resemble real data. Any deviation from the $\mathbf{x}$-distribution is exaggerated by the transformation and thus less likely to happen for NDM's predictions.

In Appendix D we provide additional samples for terminal and intermediate timesteps.

4.3. Image generation

Next, we evaluate NDMs with learnable transformations quantitatively. We train continuous time NDM on MNIST,

Table 3. Performance comparison of the DDPM and NDM on CIFAR-10 and ImageNet 32 datasets. We report FID scores for DDPM-style (FID) and DDIM-style (FID*) sampling procedures.

		CIFAR-10				ImageNet 32			
Steps	Model	NLL ↓	NELBO ↓	FID ↓	FID* ↓	NLL ↓	NELBO ↓	FID ↓	FID* ↓
1000	DDPM	3.11	3.18	**11.44**	**13.35**	3.89	3.95	**16.18**	**19.08**
	NDM	**3.02**	**3.03**	11.82	13.79	**3.79**	**3.82**	17.02	19.76
10	DDPM	5.02	5.13	37.83	**19.89**	6.28	6.42	53.51	**26.47**
	NDM	**4.63**	**4.74**	**31.56**	22.20	**5.81**	**5.94**	**45.38**	29.95
$1000 \to 10$	DDPM	8.78	8.98	**43.85**	17.73	10.99	11.23	**58.35**	25.53
	NDM	**8.58**	**8.81**	48.41	**16.96**	**10.78**	**11.06**	62.12	**23.77**

CIFAR-10, and downsampled ImageNet datasets. Table 2 summarizes our results, reporting NLL. NDMs demonstrates performance on CIFAR-10 that is comparable with the baselines and outperforms baselines on ImageNet.

Then, we compare NDM with the DDPM baseline on MNIST, CIFAR-10, and ImageNet 32 datasets. To ensure a fair comparison, when implementing DDPM we use an NDM with fixed identity transformation $F_\varphi(\mathbf{x}, t) = \mathbf{x}$. Therefore, we train both models with the same objective (8) and hyperparameters. The first part of Table 3 summarizes our results, reporting NLL, NELBO (8), and FID score. NDM demonstrates comparable sample quality with the baseline on all datasets and consistently outperforms the baseline on NLL and NELBO, especially for smaller numbers of steps. This improvement may be attributed to NDM's ability to fit distributions of the forward process and simplify the denoising task for the reverse process.

We also compare NDM with DDPM in a setup where we train both models with $T = 1000$ steps and then sample with fewer steps. The second part of Table 3 summarizes our results, which are consistent with the corresponding numbers of steps used during training. However, in absolute values, both models show worse performance when we decrease the number of steps, and NDM demonstrates a more severe degradation. This observation is especially noticeable for small numbers of steps, such as $T = 10$, where NDM has a better FID score than DDPM when trained with 10 steps, but a worse FID score when the number of steps is decreased from 1000 to 10. From this, we conclude that although NDM can in principle work with reduced number of steps it is less robust to such modifications compared to DDPM.

Finally, we demonstrate that NDMs may be successfully combined with LSGM (Vahdat et al., 2021). For this experiment we replaced the linear diffusion in the LSGM baseline for CelebA-HQ-256 with NDMs featuring the learnable F_φ. We parameterise F_φ with the same neural network architecture as baseline's architecture for parameterisation of diffusion. Table 4 demonstrates that NDMs have better likelihood estimation and sample quality.

Table 4. Generative results on CelebA-HQ-256 for LSGM and NDM with learnable transformations in the latent space of VAE.

Model	NLL ↓	FID ↓
LSGM (Vahdat et al., 2021)	≤ 0.70	7.22
Latent NDM (ours)	$\leq \mathbf{0.65}$	**7.18**

In Appendix E we provide a proof of concept experiment, which demonstrates that we can learn simpler generative dynamics compared to conventional diffusion models. In this experiment we restrict the reverse process to learn dynamic optimal transport trajectories. It is not possible to match such a reverse process with a predefined forward process, but NDM allows to capture the data distribution with the simpler generative dynamics.

See Appendix D for additional results and ablation studies.

5. Related work

NDMs build on diffusion probabilistic models originally proposed by Sohl-Dickstein et al. (2015), which can be considered as an instance of (hierarchical) variational autoencoders (VAEs) (Kingma & Welling, 2014; Rezende et al., 2014). Recently, the theory of diffusion models was extended to deterministic sampling (Song et al., 2021a) and continuous time (Song et al., 2021c). These results allowed to reach impressive performance in image generation tasks (Ho et al., 2020; Song et al., 2021c; Dhariwal & Nichol, 2021; Kingma et al., 2021). However, most existing diffusion models have a significant limitation in that they rely on a pre-specified and simple noise injection process that is unable to adapt to the specific task or data at hand. To overcome this, researchers have explored ways to generalize diffusion models.

Various papers have since proposed ways to speed up sam-

pling from diffusion models. Tachibana et al. (2021) and Liu et al. (2022) proposed alternative SDE and ODE solvers. Xiao et al. (2021) proposed replacing simple Gaussian distributions at each generation step with distributions learned by GANs (Goodfellow et al., 2014). Some works proposed methods like iterative distillation with a reduction in the number of steps (Salimans & Ho, 2022) and iterative straightening of trajectories (Liu et al., 2023; Liu, 2022). While these methods change the generative process, they are compatible with NDMs.

Several papers proposed constructing the process of data corruption not by noise injection, but rather by blurring (Rissanen et al., 2023; Daras et al., 2022; Hoogeboom & Salimans, 2023) or through another linear transformation (Singhal et al., 2023). Another line of work modifies directly the dynamics of diffusion models through mapping the data into the latent space of VAE (Vahdat et al., 2021; Rombach et al., 2022), hierarchical VAE (Gu et al., 2023) or normalizing flow (Kim et al., 2022) models and then runs standard linear diffusion. As we demonstrate in Tables 1, these arise as distinct special cases of NDMs for specific choices of the transformation F_φ.

In another line of works (De Bortoli et al., 2021; Wang et al., 2021; Peluchetti), finite-time diffusion constructions were proposed using diffusion bridge theory to address the approximation error incurred by infinite-time denoising constructions. While such approaches allow learning forward transformations, they require inferring all latent variables for each optimization step. This limitation break the simulation-free paradigm and can make these models expensive to train. NDM in contrast allows learning forward transformations efficiently and simulation-free.

Inspired by diffusion models, several works (Lipman et al., 2023; Neklyudov et al., 2022) have proposed simulation-free objectives for training continuous normalizing flows. These approaches are similar to diffusion models as they rely on the idea of reversing a predefined corruption process. Later, some works (Albergo & Vanden-Eijnden, 2023; Lee et al., 2023) extended these ideas and proposed to learn the forward process. However, although NDMs and these works are similar in spirit, they differ in that they optimize the forward process specifically to obtain straight generative trajectories, while in our approach we optimize learnable forward process to minimize variational bound on NLL, which not necessarily leads to straight generative trajectories.

In concurrent and independent work, Nielsen et al. (2024) introduced DiffEnc, also adding time-dependent transformations in diffusion models. While the underlying idea is similar, there are some differences between DiffEnc and NDM. The two approaches utilize different parameterisations and noise injection schedules. In addition, DiffEnc approximates time derivatives of the data transformations leading to biased stochastic gradients, while NDM calculates exact time derivatives using JVP.

In Appendix B, we provide further discussion, details and comparisons with related works.

6. Limitations

Compared to conventional diffusion models, NDMs with learnable transformations have twice as many parameters, which results in slower training. Specifically, in experiments on images, NDMs with learnable transformations take approximately 2.3 times longer than DDPM to train. However, no additional techniques where necessary to ensure stable training of NDMs. Additionally, in Appendix D, we provide an ablation study demonstrating that performance improvements are not achieved by increasing the number of parameters.

Another distinction between NDMs and DDPM is the importantance for NDMs in using the full objective (8) when training the model. A simplified objective, such as $\mathcal{L}_{\text{simple}}$ used in DDPM, which measures how well the model predicts injected noise and does not take into account the transformation F_φ, can cause the collapse of this transformation to 0. The reason for this is that it becomes trivial to identify the injected noise through $\mathbf{z}_t$.

Finally, unlike conventional diffusion, the generative process of NDMs with learnable transformations depends on the parameters of the forward process. Therefore, in the case of learnable parameters, NDMs do not support conditional generation techniques with classifier guidance (Dhariwal & Nichol, 2021). However, we can utilize alternative approaches (Wu et al., 2023) to enable conditional generation from NDMs, but we will defer this to future research.

7. Conclusion

We introduced Neural Diffusion Models (NDMs), a new class of diffusion models that enables defining and learning the general forward noising process. First, we showed how to optimize NDMs using a variational bound in a simulation-free setting. Then, we derived a time-continuous formulation of NDMs allowing for fast and reliable inference and likelihood evaluation using off-the-shelf numerial ODE and SDE solvers. Next, we demonstrated how some existing diffusion models appear as a special cases of NDMs. For NDMs with learnable transformations we studied their utility on standard image generation benchmarks. NDMs significantly outperforms conventional diffusion models in terms of likelihood, achieving state-of-the-art results for ImageNet and CelebA-HQ, and produces samples of comparable or better quality.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Albergo, M. S. and Vanden-Eijnden, E. Building normalizing flows with stochastic interpolants. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=li7qeBbCR1t.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Chen, T., Liu, G.-H., and Theodorou, E. A. Likelihood training of schr\" odinger bridge using forward-backward sdes theory. *arXiv preprint arXiv:2110.11291*, 2021.

Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., and Bharath, A. A. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65, 2018.

Dai, Z., Yang, Z., Yang, F., Cohen, W. W., and Salakhutdinov, R. R. Good semi-supervised learning that requires a bad gan. *Advances in neural information processing systems*, 30, 2017.

Daras, G., Delbracio, M., Talebi, H., Dimakis, A. G., and Milanfar, P. Soft diffusion: Score matching for general corruptions. *arXiv preprint arXiv:2209.05442*, 2022.

De Bortoli, V., Thornton, J., Heng, J., and Doucet, A. Diffusion schrödinger bridge with applications to score-based generative modeling. *Advances in Neural Information Processing Systems*, 34:17695–17709, 2021.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Deng, L. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.

Dhariwal, P. and Nichol, A. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Dormand, J. R. and Prince, P. J. A family of embedded runge-kutta formulae. *Journal of computational and applied mathematics*, 6(1):19–26, 1980.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Grathwohl, W., Chen, R. T. Q., Bettencourt, J., and Duvenaud, D. Scalable reversible generative models with free-form continuous dynamics. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJxgknCcK7.

Gu, J., Zhai, S., Zhang, Y., Bautista, M. Á., and Susskind, J. M. f-DM: A multi-stage diffusion model via progressive signal transformation. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=iBdwKIsg4m.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Hoogeboom, E. and Salimans, T. Blurring diffusion models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=OjDkC57x5sz.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

Kim, D., Na, B., Kwon, S. J., Lee, D., Kang, W., and Moon, I.-c. Maximum likelihood training of implicit nonlinear diffusion model. *Advances in Neural Information Processing Systems*, 35:32270–32284, 2022.

Kingma, D., Salimans, T., Poole, B., and Ho, J. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. *International Conference on Learning Representations*, 2014.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Lee, S., Kim, B., and Ye, J. C. Minimizing trajectory curvature of ode-based generative models. In *International Conference on Machine Learning*, pp. 18957–18973. PMLR, 2023.

Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Liu, L., Ren, Y., Lin, Z., and Zhao, Z. Pseudo numerical methods for diffusion models on manifolds. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=PlKWVd2yBkY.

Liu, Q. Rectified flow: A marginal preserving approach to optimal transport. *arXiv preprint arXiv:2209.14577*, 2022.

Liu, X., Gong, C., and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=XVjTT1nw5z.

MacKay, D. J. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

Neklyudov, K., Severo, D., and Makhzani, A. Action matching: A variational method for learning stochastic dynamics from samples. *arXiv preprint arXiv:2210.06662*, 2022.

Nichol, A. Q. and Dhariwal, P. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pp. 8162–8171. PMLR, 2021.

Nielsen, B. M. G., Christensen, A., Dittadi, A., and Winther, O. Diffenc: Variational diffusion with a learned encoder. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=8nxy1bQWTG.

Øksendal, B. and Øksendal, B. *Stochastic differential equations*. Springer, 2003.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. Normalizing flows for probabilistic modeling and inference. *The Journal of Machine Learning Research*, 22(1):2617–2680, 2021.

Peluchetti, S. Non-denoising forward-time diffusions.

Popov, V., Vovk, I., Gogoryan, V., Sadekova, T., and Kudinov, M. Grad-tts: A diffusion probabilistic model for text-to-speech. In *International Conference on Machine Learning*, pp. 8599–8608. PMLR, 2021.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pp. 1278–1286. PMLR, 2014.

Rissanen, S., Heinonen, M., and Solin, A. Generative modelling with inverse heat dissipation. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=4PJUBT9f2Ol.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Saharia, C., Ho, J., Chan, W., Salimans, T., Fleet, D. J., and Norouzi, M. Image super-resolution via iterative refinement. *arXiv preprint arXiv:2104.07636*, 2021.

Salimans, T. and Ho, J. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=TIdIXIpzhoI.

Singhal, R., Goldstein, M., and Ranganath, R. Where to diffuse, how to diffuse, and how to get back: Automated learning for multivariate diffusions. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=osei3IzUia.

Smale, S. and Hirsch, M. W. *Differential equations, dynamical systems, and linear algebra*, volume 60. Elsevier, 1974.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pp. 2256–2265. PMLR, 2015.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=St1giarCHLP.

Song, Y., Kim, T., Nowozin, S., Ermon, S., and Kushman, N. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv preprint arXiv:1710.10766*, 2017.

Song, Y., Durkan, C., Murray, I., and Ermon, S. Maximum likelihood training of score-based diffusion models. *Advances in Neural Information Processing Systems*, 34: 1415–1428, 2021b.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021c. URL https://openreview.net/forum?id=PxTIG12RRHS.

Tachibana, H., Go, M., Inahara, M., Katayama, Y., and Watanabe, Y. It\^{o}-taylor sampling scheme for denoising diffusion probabilistic models using ideal derivatives. *arXiv preprint arXiv:2112.13339*, 2021.

Tomczak, J. M. *Deep generative modeling*. Springer, 2022.

Trippe, B. L., Yim, J., Tischer, D., Baker, D., Broderick, T., Barzilay, R., and Jaakkola, T. S. Diffusion probabilistic modeling of protein backbones in 3D for the motif-scaffolding problem. In *The Eleventh International Conference on Learning Representations*, 2023.

Vahdat, A., Kreis, K., and Kautz, J. Score-based generative modeling in latent space. *Advances in Neural Information Processing Systems*, 34:11287–11302, 2021.

Van Den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. Pixel recurrent neural networks. In *International conference on machine learning*, pp. 1747–1756. PMLR, 2016.

Wang, G., Jiao, Y., Xu, Q., Wang, Y., and Yang, C. Deep generative learning via schrödinger bridge. In *International Conference on Machine Learning*, pp. 10794–10804. PMLR, 2021.

Watson, J. L., Juergens, D., Bennett, N. R., Trippe, B. L., Yim, J., Eisenach, H. E., Ahern, W., Borst, A. J., Ragotte, R. J., Milles, L. F., et al. Broadly applicable and accurate protein design by integrating structure prediction networks and diffusion generative models. *bioRxiv*, pp. 2022–12, 2022.

Wu, L., Trippe, B. L., Naesseth, C. A., Blei, D. M., and Cunningham, J. P. Practical and asymptotically exact conditional sampling in diffusion models. *arXiv preprint arXiv:2306.17775*, 2023.

Xiao, Z., Kreis, K., and Vahdat, A. Tackling the generative learning trilemma with denoising diffusion GANs. *arXiv preprint arXiv:2112.07804*, 2021.

Yang, L., Zhang, Z., Song, Y., Hong, S., Xu, R., Zhao, Y., Shao, Y., Zhang, W., Cui, B., and Yang, M.-H. Diffusion models: A comprehensive survey of methods and applications. *arXiv preprint arXiv:2209.00796*, 2022.

A. Derivations and proofs

A.1. Forward posterior

First, we rewrite the marginal distribution (6) in terms of standard normally distributed ε_t, ε_s for s and t, where $s < t$:

$$\mathbf{z}_t = \alpha_t F_\varphi(\mathbf{x}, t) + \sigma_t \varepsilon_t, \tag{12}$$

$$\mathbf{z}_s = \alpha_s F_\varphi(\mathbf{x}, s) + \sigma_s \varepsilon_s. \tag{13}$$

Next, we constructively introduce the posterior distribution $q_\varphi(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x})$. To sample $\mathbf{z}_s$ given $\mathbf{z}_t$ and $\mathbf{x}$ while preserving the correct marginal distribution $q_\varphi(\mathbf{z}_s|\mathbf{x})$, we can combine the noise ε_t with additional noise $\tilde{\varepsilon}_{s|t}$ as follows:

$$\mathbf{z}_s = \alpha_s F_\varphi(\mathbf{x}, s) + \sqrt{\sigma_s^2 - \tilde{\sigma}_{s|t}^2}\varepsilon_t + \tilde{\sigma}_{s|t}\tilde{\varepsilon}_{s|t}. \tag{14}$$

The samples $\mathbf{z}_s$ follow a (conditional) normal distribution. By marginalizing ε_t and $\tilde{\varepsilon}_{s|t}$, we obtain a normal distribution with mean $\alpha_s F\varphi(\mathbf{x}, s)$ and variance $\sigma_s^2 - \tilde{\sigma}_{s|t}^2 + \tilde{\sigma}_{s|t}^2 = \sigma_s^2$. Therefore, this sampling procedure satisfies $q_\varphi(\mathbf{z}_s|\mathbf{x}) = \int q_\varphi(\mathbf{z}_t|\mathbf{x}) q_\varphi(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x}) d\mathbf{z}_t$.

The equation (14) relies on ε_t, which we do not have explicit access to. However, once we know $\mathbf{z}_t$ and $\mathbf{x}$, we can calculate it from (12) as $\varepsilon_t = \frac{\mathbf{z}_t - \alpha_t F_\varphi(\mathbf{x}, t)}{\sigma_t}$ and substitute it in (14):

$$\mathbf{z}_s = \alpha_s F_\varphi(\mathbf{x}, s) + \frac{\sqrt{\sigma_s^2 - \tilde{\sigma}_{s|t}^2}}{\sigma_t}\Big(\mathbf{z}_t - \alpha_t F_\varphi(\mathbf{x}, t)\Big) + \tilde{\sigma}_{s|t}\tilde{\varepsilon}_{s|t}. \tag{15}$$

Using this constructive definition, we obtain the posterior distribution (7).

A.2. Objective

To calculate the diffusion term $\mathcal{L}_{\text{diff}}$ (9) of the objective, we need to compute the KL divergence between the forward posterior distribution $q_\varphi(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x})$ and the reverse distribution $p_\theta(\mathbf{z}_s|\mathbf{z}_t)$. Since we use parameterization $p_\theta(\mathbf{z}_s|\mathbf{z}_t) = q_\varphi(\mathbf{z}_s|\mathbf{z}_t, \hat{\mathbf{x}}_\theta(\mathbf{z}_t, t))$, both of these distributions are normal distributions with the same variance, so we can evaluate the KL divergence between them analytically as follows:

$$\begin{aligned}
&\mathrm{D}_{\mathrm{KL}}\Big(q_\varphi(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x})||p_\theta(\mathbf{z}_s|\mathbf{z}_t)\Big) = \\
&= \frac{1}{2\tilde{\sigma}_{s|t}^2}\Bigg\|\alpha_s F_\varphi(\mathbf{x}, s) + \frac{\sqrt{\sigma_s^2 - \tilde{\sigma}_{s|t}^2}}{\sigma_t}\Big(\cancel{\mathbf{z}_t} - \alpha_t F_\varphi(\mathbf{x}, t)\Big) - \\
&\qquad \alpha_s F_\varphi(\hat{\mathbf{x}}_\theta(\mathbf{z}_t, t), s) - \frac{\sqrt{\sigma_s^2 - \tilde{\sigma}_{s|t}^2}}{\sigma_t}\Big(\cancel{\mathbf{z}_t} - \alpha_t F_\varphi(\hat{\mathbf{x}}_\theta(\mathbf{z}_t, t), t)\Big)\Bigg\|_2^2 \qquad (16) \\
&= \frac{1}{2\tilde{\sigma}_{s|t}^2}\Bigg\|\alpha_s\Big(F_\varphi(\mathbf{x}, s) - F_\varphi(\hat{\mathbf{x}}_\theta(\mathbf{z}_t, t), s)\Big) + \\
&\qquad \frac{\sqrt{\sigma_s^2 - \tilde{\sigma}_{s|t}^2}}{\sigma_t}\alpha_t\Big(F_\varphi(\hat{\mathbf{x}}_\theta(\mathbf{z}_t, t), t) - F_\varphi(\mathbf{x}, t)\Big)\Bigg\|_2^2. \qquad (17)
\end{aligned}$$

With a learnable transformation F_φ, the term $\mathcal{L}_{\text{prior}}$ becomes dependent on the parameters φ, necessitating its optimization

during training. We can compute the prior term as follows:

$$\mathrm{D_{KL}}\Big(q_\varphi(\mathbf{z}_T|\mathbf{x})||p(\mathbf{z}_T)\Big) = \frac{1}{2}\left[\log\frac{|I|}{|\sigma_T^2 I|} - d + Tr\{I^{-1}\sigma_t^2 I\} + \Big\|0 - \alpha_T F_\varphi(\mathbf{x},T)\Big\|_2^2\right] \tag{18}$$

$$= \frac{1}{2}\left[-d\log\sigma_T^2 - d + d\sigma_T^2 + \Big\|\alpha_T F_\varphi(\mathbf{x},T)\Big\|_2^2\right] \tag{19}$$

$$= \frac{1}{2}\left[d\Big(\sigma_T^2 - \log\sigma_T^2 - 1\Big) + \alpha_T^2\Big\|F_\varphi(\mathbf{x},T)\Big\|_2^2\right]. \tag{20}$$

Here, d represents the dimensionality of the data space.

A.3. Reverse SDE and ODE

As discussed in Section 3.2, when the number of steps, denoted as T, tends to infinity for NDM, we can switch to continuous time.

In the discrete time setting, we define the time step as $t \in [0, 1, \dots, T]$. In the continuous time setting, we utilize the unit interval, denoting time as $t \in [0, 1]$. Nevertheless, for the sake of notational simplicity in this and subsequent sections, we will consider the discrete time to also lie within the unit interval, with $t \in [\frac{0}{T}, \frac{1}{T}, \dots, \frac{T}{T}]$.

To derive the stochastic differential equation (SDE) for the reverse process $p_\theta(\mathbf{z}_s|\mathbf{z}_t)$ in NDM, we first obtain an SDE that depends on the data point $\mathbf{x}$ and whose solution corresponds to the posterior distribution $q_\varphi(\mathbf{z}_{t-\Delta t}|\mathbf{z}_t, \mathbf{x})$. By defining $p_\theta(\mathbf{z}_s|\mathbf{z}_t)$ through $q_\varphi(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x})$ with the prediction $\hat{\mathbf{x}}_\theta(\mathbf{z}_t, t)$ instead of $\mathbf{x}$, we can subsequently replace the prediction $\hat{\mathbf{x}}_\theta(\mathbf{z}_t, t)$ and derive the SDE for the reverse process.

We constructively derive the SDE for the posteriors $q_\varphi(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x})$. First, let us consider the following auxiliary SDE with backward time flow:

$$d\varepsilon_t = \frac{\dot{\nu}_t}{2}\varepsilon_t dt + \sqrt{\dot{\nu}_t}d\mathbf{w}. \tag{21}$$

It is straightforward to show that the solution to this SDE corresponds to the following distribution:

$$q(\varepsilon_s|\varepsilon_t) = \mathcal{N}(\varepsilon_s; \sqrt{1-\bar{\sigma}_{s|t}^2}\varepsilon_t; \bar{\sigma}_{s|t}^2 I), \quad \text{where} \quad \bar{\sigma}_{s|t}^2 = 1 - e^{\nu_s - \nu_t}. \tag{22}$$

To derive the SDE for the posteriors $q_\varphi(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x})$, we can apply the following function to both the SDE (21) and the distribution (22):

$$G(\mathbf{x}, \varepsilon_t, t) = \alpha_t F_\varphi(\mathbf{x}, t) + \sigma_t \varepsilon_t \tag{23}$$

Note that after applying the function G, the distribution (22) matches the posterior distribution $q_\varphi(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x})$. Therefore, the desired SDE for $q_\varphi(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x})$ is obtained by transforming the SDE (21) using Ito's formula (Øksendal & Øksendal, 2003):

$$d\mathbf{z}_t = \left[\frac{\partial G(\mathbf{x},\varepsilon,t)}{\partial t}\Big|_{\varepsilon=\varepsilon_t} + \frac{\dot{\nu}_t}{2}\frac{\partial G(\mathbf{x},\varepsilon_t,t)}{\partial \varepsilon_t}\varepsilon_t - \frac{\dot{\nu}_t}{2}\frac{\partial^2 G(\mathbf{x},\varepsilon_t,t)}{\partial \varepsilon_t^2}\right]dt + \sqrt{\dot{\nu}_t}\frac{\partial G(\mathbf{x},\varepsilon_t,t)}{\partial \varepsilon_t}d\mathbf{w} \tag{24}$$

$$= \left[\dot{\alpha}_t F_\varphi(\mathbf{x},t) + \alpha_t \dot{F}_\varphi(\mathbf{x},t) + \dot{\sigma}_t\varepsilon_t + \frac{\dot{\nu}_t}{2}\sigma_t\varepsilon_t\right]dt + \sqrt{\dot{\nu}_t}\sigma_t d\mathbf{w} \tag{25}$$

$$= \left[\frac{\dot{\alpha}_t}{\alpha_t}(\mathbf{z}_t - \sigma_t\varepsilon_t) + \alpha_t\dot{F}_\varphi(\mathbf{x},t) + \dot{\sigma}_t\varepsilon_t + \frac{\dot{\nu}_t}{2}\sigma_t\varepsilon_t\right]dt + \sqrt{\dot{\nu}_t}\sigma_t d\mathbf{w} \tag{26}$$

$$= \left[\alpha_t\dot{F}_\varphi(\mathbf{x},t) + \frac{\partial \log\alpha_t}{\partial t}\mathbf{z}_t - \frac{1}{2}\left(\frac{\partial\sigma_t^2}{\partial t} - 2\frac{\partial\log\alpha_t}{\partial t}\sigma_t^2 + \dot{\nu}_t\sigma_t^2\right)\left(-\frac{\varepsilon_t}{\sigma_t}\right)\right]dt + \sqrt{\dot{\nu}_t}\sigma_t d\mathbf{w} \tag{27}$$

$$= \left[\alpha_t\dot{F}_\varphi(\mathbf{x},t) + r(t)\mathbf{z}_t - \frac{1}{2}\left(\frac{\partial\sigma_t^2}{\partial t} - 2r(t)\sigma_t^2 + \dot{\nu}_t\sigma_t^2\right)s(\mathbf{x},\mathbf{z}_t,t)\right]dt + \sqrt{\dot{\nu}_t}\sigma_t d\mathbf{w}, \tag{28}$$

$$\text{where} \quad r(t) = \frac{\partial\log\alpha_t}{\partial t} \quad \text{and} \quad s(\mathbf{x},\mathbf{z}_t,t) = \frac{\alpha_t F_\varphi(\mathbf{x},t) - \mathbf{z}_t}{\sigma_t^2} \tag{29}$$

To obtain the SDE for the reverse process, we can substitute the prediction $\hat{\mathbf{x}}_\theta(\mathbf{z}_t, t)$ instead of $\mathbf{x}$. This substitution yields the SDE (11):

$$d\mathbf{z}_t = \left[\alpha_t \dot{F}_\varphi(\hat{\mathbf{x}}_\theta(\mathbf{z}_t, t), t) + r(t)\mathbf{z}_t - \frac{1}{2}\left(\frac{\partial \sigma_t^2}{\partial t} - 2r(t)\sigma_t^2 + \dot{\nu}_t \sigma_t^2\right) s_\theta(\mathbf{z}_t, t)\right] dt + \sqrt{\dot{\nu}_t}\sigma_t d\mathbf{w}, \tag{30}$$

$$\text{where} \quad s_\theta(\mathbf{z}_t, t) = \frac{\alpha_t F_\varphi(\hat{\mathbf{x}}_\theta(\mathbf{z}_t, t), t) - \mathbf{z}_t}{\sigma_t^2} \tag{31}$$

As discussed earlier, we can leverage the Jacobian-Vector Product (JVP) trick (Smale & Hirsch, 1974) to calculate $\dot{F}_\varphi$.

In the case where ν_t is a constant, the dynamics become deterministic and can be described by ordinary differential equations (ODEs). In our experiments, we utilize these ODEs to model the generative process as a continuous normalizing flow (Chen et al., 2018; Grathwohl et al., 2019) and estimate densities.

A.4. Continuous time objective

When we switch to continuous time, the discrete objective (9) transforms from finite sum of KL divergances into integral, which we can easily derive as soon as we have access to both stochastic differential equations associated with the forward process (28) and with the rewerse process (30). In continuous time the diffusion term $\mathcal{L}_{\text{diff}}$ (10) is equal to:

$$\begin{aligned}\mathcal{L}_{\text{diff}} = \int_0^1 \frac{1}{g^2(t)} \Bigg\| & \alpha_t \Big(\dot{F}_\varphi(\mathbf{x}, t) - \dot{F}_\varphi(\hat{\mathbf{x}}_\theta(\mathbf{z}_t, t), t)\Big) + \\ & \frac{1}{2}\left(\frac{\partial \sigma_t^2}{\partial t} - 2r(t)\sigma_t^2 + g^2(t)\right)\Big(s(\mathbf{x}, \mathbf{z}_t, t) - s(\hat{\mathbf{x}}_\theta(\mathbf{z}_t, t), \mathbf{z}_t, t)\Big)\Bigg\|_2^2 dt,\end{aligned}$$

$$\text{where} \quad r(t) = \frac{\partial \log \alpha_t}{\partial t}, \quad g^2(t) = \dot{\nu}_t \sigma_t^2 \quad \text{and} \quad s(\mathbf{x}, \mathbf{z}_t, t) = \frac{\alpha_t F_\varphi(\mathbf{x}, t) - \mathbf{z}_t}{\sigma_t^2}. \tag{32}$$

As we can see, these equation contains $\dot{F}_\varphi$ as a component. In general, we do not have explicit access to the time derivative of the forward transformation F_φ. However, we will focus on cases where the forward transformation is differentiable. By utilizing automatic differentiation tools, we can calculate the time derivatives of F_φ. Nevertheless, when $\mathbf{x}$ is fixed, the function $F_\varphi(\mathbf{x}, \cdot)$ becomes a scalar-to-vector function. To compute its time derivative using simple backpropagation, we would need to execute it for all outputs of F_φ, resulting in quadratic computational complexity. Fortunately, there exists a more efficient method to obtain the time derivative, the Jacobian-Vector Product trick (Smale & Hirsch, 1974). The Jacobian of the transformation function with $\mathbf{x}$ fixed is represented as a column matrix. Therefore, by computing the product of the Jacobian with a one-dimensional vector, we can obtain a vector of time derivatives.

B. Connections with other works

We introduce NDMs as a comprehensive framework that generalises various existing approaches. Here we provide Table 5 which is an extended version of Table 1, that demonstrates how existing approaches appear as a spatial cases of NDMs.

We also provide an extended discussion on the connection between NDM and other related works.

B.1. Diffusion in latent space

The concept of a learnable forward process is not entirely new. In some sense models that run a diffusion process in the latent space of a VAE (Vahdat et al., 2021; Rombach et al., 2022), a hierarchical VAE (Gu et al., 2023), or a Flow model (Kim et al., 2022) can be viewed as diffusion models with a learnable forward process. These models optimize the mapping to the latent space. Consequently, projecting the diffusion generative dynamic from the latent to the data space introduces a novel, nonlinear, and learnable generative dynamic. However, these models still rely on conventional diffusion in the latent space.

Additionally, these models can be viewed as spatial cases of NDM with a specific choice of the transformation $F_\varphi(\mathbf{x}, t)$.

Table 5. Summary of existing diffusion models as instances of Neural Diffusion Models (NDM).

Model	Distribution $q(\mathbf{z}_t \mid x)$	NDM's $F(\mathbf{x}, t)$	Comment
DDPM (Ho et al., 2020) / DDIM (Song et al., 2021a)	$\mathcal{N}\Big(\mathbf{z}_t; \alpha_t \mathbf{x}, \sigma_t^2 I\Big)$	$\mathbf{x}$	
Flow Matching OT (Lipman et al., 2023)	$\mathcal{N}\Big(\mathbf{z}_t; \alpha_t \mathbf{x}, \sigma_t^2 I\Big)$	$\mathbf{x}$	$\alpha_t = t$, $\sigma_t = 1 - (1 - \sigma_{\min})t$
VDM (Kingma et al., 2021)	$\mathcal{N}\Big(\mathbf{z}_t; \alpha_t \mathbf{x}, \sigma_t^2 I\Big)$	$\mathbf{x}$	$\alpha_t^2 = \mathrm{sigmoid}(-\gamma_\eta(t))$, $\sigma_t^2 = \mathrm{sigmoid}(\gamma_\eta(t))$
IHDM (Rissanen et al., 2023)	$\mathcal{N}\Big(\mathbf{z}_t; Ve^{-\Lambda t}V^T\mathbf{x}, \sigma^2 I\Big)$	$Ve^{-\Lambda t}V^T\mathbf{x}$	$\alpha_t = 1, \sigma_t = \sigma$, σ is fixed
Blurring Diffusion (Hoogeboom & Salimans, 2023)	$\mathcal{N}\Big(\mathbf{z}_t; \alpha_t e^{-\Lambda t}V^T\mathbf{x}, \sigma_t^2 I\Big)$	$e^{-\Lambda t}V^T\mathbf{x}$	$p(x \mid z_0) = \mathcal{N}\Big(x; aVz_0, \sigma\Big)$
Soft Diffusion (Daras et al., 2022)	$\mathcal{N}\Big(\mathbf{z}_t; C_t\mathbf{x}, s_t^2 I\Big)$	$C_t\mathbf{x}$	$\alpha_t = 1, \sigma_t^2 = s_t^2$
LSGM (Vahdat et al., 2021)	$\mathcal{N}\Big(\mathbf{z}_t; \alpha_t E(\mathbf{x}), \sigma_t^2 I\Big)$	$E(\mathbf{x})$	$p(x \mid z_0) = \mathcal{N}\Big(x; aD(z_0), \sigma^2\Big)$
f-DM (Gu et al., 2023)	$\mathcal{N}\Big(\mathbf{z}_t; \alpha_t \mathbf{x}_t, \sigma_t^2 I\Big)$	$\mathbf{x}_t = \frac{(t-\tau_k)\hat{x}^k + (\tau_{k+1}-t)x^k}{\tau_{k+1}-\tau_k}$, where $\tau_k \leq t < \tau_{k+1}$	$x^k = f_{0:k}(x)$ $\hat{x}^k = \begin{cases} g_k(f_{k+1}(x^k)), & \text{if } k < K, \\ x^k, & \text{if } k = K. \end{cases}$

For example, F_φ might be selected as the VAE's time independent encoder in the case of (Vahdat et al., 2021) or the time independent Flow model in the case of (Kim et al., 2022).

B.2. Schrödinger Bridges

Another line of works (De Bortoli et al., 2021; Wang et al., 2021; Peluchetti; Chen et al., 2021) are approaches based on Schrödinger Bridge theory. While such approaches allow learning forward transformations, in contrast to NDM, these approaches are not simulation-free. In Schrödinger Bridge models, we typically lack direct access to the distribution $q(\mathbf{z}_t|\mathbf{x})$. Consequently, to sample the latent variable $\mathbf{z}_t$ in training time, we must simulate the full stochastic process, such as the stochastic differential equations. This characteristic makes Schrödinger Bridge models expensive in training and not simulation-free.

In contrast, NDM framework, by design, has access to $q(\mathbf{z}_t|\mathbf{x})$. Thus, with NDM, when training a model with T time steps, there is no need to propagate F_φ for T times at each step of the training procedure. Instead, the NDM framework enables sampling of the intermediate latent variables $\mathbf{z}_t$ directly from the distribution $q(\mathbf{z}_t|\mathbf{x})$. Therefore, we can maintain the training paradigm outlined in Section 2. Instead of computing all T KL divergences for each time step, we can approximate the objective using the Monte Carlo method by calculating just one KL divergence for a uniformly sampled time step $t \in [1; T]$, as described in Algorithm 1.

This approach allows us to train the model with batches of shape $[batch_size, d]$ rather than $[batch_size, T, d]$. Consequently, NDM can leverage larger batch sizes and use just one call of F_φ for inferring latent variables $\mathbf{z}_t$.

B.3. Stochastic Interpolants

Albergo & Vanden-Eijnden (2023) proposed a Stochastic Interpolant approach, which provide more flexibility then conventional diffusion models in defining and even learning of the forward process. While we find stochastic interpolants intriguing and promising, as well as related to our work, these methods differ significantly.

Firstly, stochastic interpolants represent an approach to learning continuous-time deterministic generative dynamics, whereas NDM learns stochastic dynamics in either discrete or continuous time, which can subsequently may be transformed into a deterministic process.

Secondly, in NDM, the model is trained by optimizing the variational bound on the likelihood, while Stochastic Interpolants are trained by optimizing the generalization of the Flow Matching objective (Lipman et al., 2023).

Lastly, NDM joint learns both the forward and reverse processes by optimizing the likelihood, whereas stochastic interpolants

learn the generative process with a fixed forward process. Albergo & Vanden-Eijnden (2023) demonstrate the possibility of constructing an optimization procedure for the forward process through a max-min game to solve a dynamic optimal transport problem. However, the purpose of this optimization differs from that of NDM.

Moreover, max-min optimization, as employed in Stochastic Interpolants, is notably less stable compared to min-min optimization in NDM. Additionally, Stochastic Interpolants do not present experimental results for the optimization of the forward process.

B.4. DiffEnc

In concurrent work, Nielsen et al. (2024) introduced DiffEnc. DiffEnc also proposes to add a time-dependent transformation to the data in the diffusion model. However, there are some distinctions between these two methods. Firstly, in NDM, we parameterize the reverse process by predicting the data point $\mathbf{x}$, while in DiffEnc, they predict the transformed data point $F_\varphi(\mathbf{x}, t)$.

Secondly, in NDM, we employ a Signal-to-Noise Ratio (SNR) schedule for noise injection from DDPM (Ho et al., 2020) and a straightforward parameterization of the model $\hat{\mathbf{x}}_\theta(\mathbf{z}_t, t)$ through predicting the injected epsilon, as detailed in Appendix C.2. Simultaneously, in DiffEnc, the authors use a learnable SNR schedule (Kingma et al., 2021) and a v-parameterization (Salimans & Ho, 2022) of $\hat{\mathbf{x}}_\theta(\mathbf{z}_t, t)$.

Finally, DiffEnc utilizes approximations of the time derivatives of data transformations F_φ, while in the NDM framework, we propose calculating exact time derivatives using Jacobian-Vector Products.

C. Implementation details

All our experiments were conducted using synthetic 2D datasets and image datasets: MNIST (Deng, 2012), CIFAR-10 (Krizhevsky et al., 2009), downsampled ImageNet (Deng et al., 2009; Van Den Oord et al., 2016) and CelebA-HQ-256 (Karras et al., 2017). For CIFAR-10 and ImageNet datasets we applied center cropping and resizing. For synthetic data, we employed a 5-layer MLP with 512 neurons in each layer, while for the images, we utilized the U-Net architecture from Dhariwal & Nichol (2021). In our experiments both the DDPM and NDM approaches were trained on identical architectures, with the same hyper-parameters and for the same number of epochs. The hyper-parameters are presented in Table 6. In experiment where we report results for the continuous time models we use importance sampling of time (Song et al., 2021b) instead of uniform sampling.

We trained models using the Adam optimizer, setting the following parameters: $\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay of 0.0, and $\varepsilon = 10^{-8}$. To facilitate the training process, we employed a polynomial decay learning rate schedule, which includes a warm-up phase for a specified number of training steps. During the warm-up phase, the learning rate is linearly increased from 10^{-8} to the peak learning rate. Once the peak learning rate is reached, the learning rate is linearly decayed to 10^{-8} until the final training step. The training was performed using Tesla V100 GPUs.

C.1. Dequantization

When reporting negative log-likelihood, we dequantize using the standard uniform dequantization. We report an importance-weighted estimate using

$$\log \frac{1}{K} \sum_{k=1}^{K} p_\theta(\mathbf{x} + u_k), \quad \text{where} \quad u_k \sim \mathcal{U}(0, 1), \tag{33}$$

with $\mathbf{x} \in [0, \dots, 255]$.

C.2. Parameterization

In order to simplify the derivations above, we have utilized the notation $\hat{\mathbf{x}}_\theta(\mathbf{z}_t, t)$ to represent the prediction of the reverse process. However, prior research has shown that predicting the injected noise ε_t can lead to improved results (Ho et al., 2020; Nichol & Dhariwal, 2021; Dhariwal & Nichol, 2021). Therefore, in all the experiments, we opt for the following

Table 6. Training hyper-parameters.

	CIFAR-10	ImageNet 32	ImageNet 64
Channels	256	256	192
Depth	2	3	3
Channels multipliers	1,2,2,2	1,2,2,2	1,2,3,4
Heads	4	4	4
Heads Channels	64	64	64
Attention resolution	16	16,8	32,16,8
Dropout	0.0	0.0	0.0
Effective Batch size	256	1024	2048
GPUs	2	4	16
Epochs	1000	200	250
Iterations	391k	250k	157k
Learning Rate	4e-4	1e-4	1e-4
Learning Rate Scheduler	Polynomial	Polynomial	Constant
Warmup Steps	45k	20k	-

parameterization:

$$\hat{\mathbf{x}}_\theta(\mathbf{z}_t, t) = \frac{\mathbf{z}_t - \sigma_t \hat{\varepsilon}_\theta(\mathbf{z}_t, t)}{\alpha_t}. \tag{34}$$

It is worth noting that with this parameterization, $\hat{\varepsilon}_\theta(\mathbf{z}_t, t)$ does not necessarily approximate the true injected noise ε_t, since this reparameterization does not account for the transformation F_φ. We believe that better parameterizations may exist for NDM, but we leave this for future research.

Furthermore, we restrict the transformation F_φ to an identity transformation for $t = 0$ through the following construction:

$$F_\varphi(\mathbf{x}, t) = (1 - t)\mathbf{x} + t\bar{F}_\varphi(\mathbf{x}, t). \tag{35}$$

This ensures that $q(\mathbf{z}_0|\mathbf{x}) \approx \delta(\mathbf{z}_0 - \mathbf{x})$, and thus also removes the need to optimize the reconstruction term $\mathcal{L}_{\text{rec}}$.

Finally, to ensure consistency with Ho et al. (2020) we use $\tilde{\sigma}^2_{s|t} = \left(\sigma_t^2 - \frac{\alpha_t^2}{\alpha_s^2}\sigma_s^2\right)\frac{\sigma_s^2}{\sigma_t^2}$ for the forward process (7). This choice of $\tilde{\sigma}^2_{s|t}$ guaranties consistency between the NDM and DDPM forward processes. For α_t and σ_t^2 we use the DDPM schedule of noise injection.

C.3. Diffusion in latent space

For experiment with diffusion in the latent space of VAE on CelebA-HQ-256, we followed LSGM (Vahdat et al., 2021) experiment setup. The only difference between LSGM baseline and our model is that we utilize learnable transformations F_φ according to NDMs framework. We apply the same hyperparameters, as LSGM.

D. Additional results

D.1. Additional evaluation

Here we provide Table 7 which contains additional resalts to Table 3. This table compare DDPM and NDMs with learnable transformations on CIFAR-10 and ImageNet 32×32 datasets with different numbers of steps.

D.2. Additional samples

In this section, we present additional illustrations showcasing the properties of NDMs.

Figure 4 provides a comparison between DDPM and NDM on a synthetic 2D data distribution. For this experiment, both models utilize $T = 10$ discrete time steps. From Figure 4c, it is evident that NDM learns to transform the data distribution.

Table 7. Performance comparison the DDPM and NDM on CIFAR-10 and ImageNet 32 datasets with different numbers of steps. We report the performance with same hyperparameters and neural networks on both models to quantify the effect of learnable transformation in fair setting. We provide likelihood (bits/dim) and negative ELBO. Additionally for CIFAR-10 and ImageNet 32 we provide FID score. Boldface numbers represent the best performance. NDM consistently outperforms in terms of NLL and NELBO with comparable sample quality to DDPM on all datasets.

		CIFAR-10			ImageNet 32		
Steps	Model	NLL ↓	NELBO ↓	FID ↓	NLL ↓	NELBO ↓	FID ↓
1000	DDPM	3.11	3.18	**11.44**	3.89	3.95	**16.18**
	NDM	**3.02**	**3.03**	11.82	**3.79**	**3.82**	17.02
100	DDPM	3.31	3.38	**11.78**	4.14	4.23	**16.66**
	NDM	**3.05**	**3.12**	11.98	**3.83**	**3.92**	17.74
50	DDPM	3.49	3.57	13.22	4.37	4.47	**18.70**
	NDM	**3.22**	**3.30**	**13.15**	**4.05**	**4.14**	18.93
10	DDPM	5.02	5.13	37.83	6.28	6.42	53.51
	NDM	**4.63**	**4.74**	**31.56**	**5.81**	**5.94**	**45.38**
1000 → 100	DDPM	3.38	3.45	**12.29**	4.23	4.32	**17.49**
	NDM	**3.30**	**3.37**	12.70	**4.15**	**4.23**	18.48
1000 → 50	DDPM	4.08	4.17	**15.24**	5.10	5.21	**20.09**
	NDM	**3.98**	**4.07**	16.83	**5.00**	**5.10**	21.11
1000 → 10	DDPM	8.78	8.98	**43.85**	10.99	11.23	**58.35**
	NDM	**8.58**	**8.81**	48.41	**10.78**	**11.06**	62.12

Additionally, after injecting noise (Figures 4a and 4d), the distributions of samples $\mathbf{z}_t$ show minimal differences between DDPM and NDM. However, when examining the predictions of data points $\hat{\mathbf{x}}_\theta(\mathbf{z}_t, t)$ (Figures 4b and 4e), NDM produces predictions that more closely resemble the true data distribution compared to DDPM.

A similar pattern emerges when applying these models to the MNIST dataset, as depicted in Figure 5. For this experiment we also use $T = 10$ discrete time steps. DDPM generates blurry predictions $\hat{\mathbf{x}}_\theta(\mathbf{z}_t, t)$ for t close to T, which bear little resemblance to real MNIST samples. Conversely, NDM produces predictions that are more similar to the true MNIST distribution, despite both models generating similar-looking noisy samples.

Finally, we include samples from both DDPM and NDM models with $T = 1000$ steps on the CIFAR-10 dataset in Figure 6. As outlined in Table 1, NDM exhibits lower sample quality based on FID measurements; however, visually there is no drop in quality.

D.3. Ablation studies

Finally, we address the question of whether the improved performance of NDM is due to the proposed method or merely the result of increasing the number of model parameters. To investigate this issue, we provide additional experiments where we double the number of DDPM parameters in two ways. The first way is to simply stack two U-Net architectures, which is the closest form to NDM. The second way is to increase the width of the U-Net architecture. Specifically, for the second way we use 384 channels instead of 256. Importantly, we left all other hyper-parameters (see Table 6), such as the learning rate and number of iterations, unchanged. As shown in Table 8, neither of these approaches yields the same results as NDM with learnable transformations. This means that the improved performance is not simply a result of the increased number of parameters.

E. Dynamic optimal transport

In this section, we present a proof-of-concept experiment demonstrating that the NDMs framework enables the learning of simpler generative trajectories. Specifically, we conduct experiments involving a 1D mixture of Gaussian distribution and dynamic optimal transport (OT).

Table 8. Comparison of NDM and DDPM with doubled number of parameters on CIFAR-10 for 10 and 1000 steps. The performance of DDPM stays the same while doubling the number of parameters, and NDM still achieves the best NLL and NELBO despite comparable number of parameters.

	10 steps			1000 steps		
Model	NLL ↓	NELBO ↓	FID ↓	NLL ↓	NELBO ↓	FID ↓
DDPM	5.02	5.13	37.83	3.11	3.18	11.44
DDPM (stack)	5.02	5.13	38.05	3.10	3.18	11.42
DDPM (wide)	5.01	5.11	37.88	3.11	3.17	**11.39**
NDM	**4.63**	**4.74**	**31.56**	**3.02**	**3.03**	11.82

While NDMs don't inherently have a direct connection with OT, we can establish a connection given the presence of infinitely many pairs of matched forward and reverse processes. This connection is facilitated by the NDMs' ability to learn the forward process. Therefore, we can consider the following setup.

We consider NDMs with a learnable function F_φ. Then, we constrain the reverse process to exclusively learn dynamic OT mappings. Finally, we train both the forward and reverse processes jointly, following the NDMs framework. In such a setup we can expect the forward process to learn such a transition from data distribution to Gaussian distribution, that aligns with the limitations imposed on the reverse process.

E.1. Restricted reverse process

To restrict the reverse process we parameterise the reverse deterministic process to have linear trajectories:

$$\mathbf{z}_t = h_\theta(t, \varepsilon) = (1-t)\hat{\mathbf{x}}_\theta(\varepsilon) + t\varepsilon, \tag{36}$$

where ε is a sample drawn from a unit Gaussian distribution. Since we are working with smooth 1D distributions, it is enough for $\hat{\mathbf{x}}_\theta$ to be monotonically increasing, so the trajectories $\mathbf{z}_t$ correspond to dynamic OT. Which means that for any parameters θ the reverse process describes dynamic OT between the standard Gaussian distribution and another distribution (not necessarily exactly the target data distribution). In practice, we parameterize $\hat{\mathbf{x}}_\theta$ using the neural network proposed by Kingma et al. (2021) for the parameterization of the Signal-to-Noise Ratio (SNR) function.

Then, we can derive an ordinary differential equation (ODE) for the reverse process:

$$d\mathbf{z}_t = \underbrace{\varepsilon - \hat{\mathbf{x}}_\theta(\varepsilon)\Big|_{\varepsilon = h_\theta^{-1}(t, \mathbf{z}_t)}}_{f_\theta(t, \mathbf{z}_t)} dt. \tag{37}$$

Next, we may switch to a stochastic differential equation (SDE) according to Song et al. (2021c):

$$d\mathbf{z}_t = \underbrace{\left[f_\theta(t, \mathbf{z}_t) - \frac{g^2(t)}{2} \nabla_{\mathbf{z}_t} \log p_\theta(\mathbf{z}_t) \right]}_{f^r_\theta(t, \mathbf{z}_t)} dt + g(t) d\bar{\mathbf{w}}. \tag{38}$$

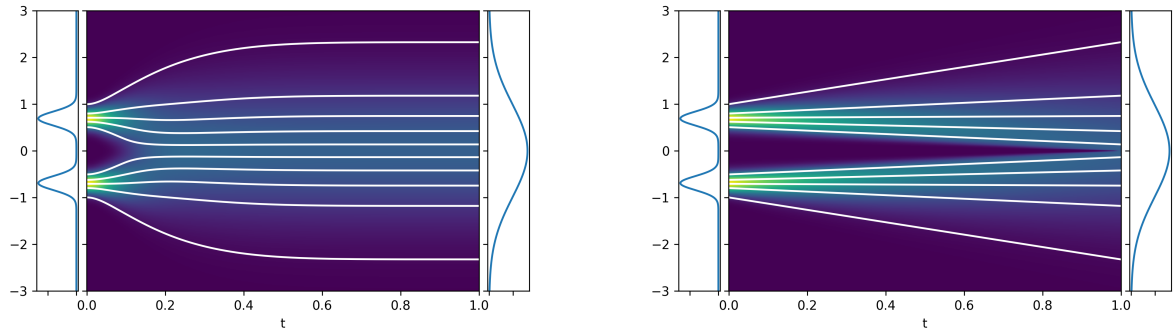
As soon as we have access to h_θ^{-1}, we may find:

$$\nabla_{\mathbf{z}_t} \log p_\theta(\mathbf{z}_t) = \nabla_{\mathbf{z}_t} \left[\log p(\varepsilon) - \log \left| \frac{\partial \mathbf{z}_t}{\partial \varepsilon} \right| \right] \Big|_{\varepsilon = h_\theta^{-1}(t, \mathbf{z}_t)} \tag{39}$$

$$= \nabla_{\mathbf{z}_t} \left[\log p(\varepsilon) - \log \left| (1-t) \frac{\partial \mathbf{x}_t}{\partial \varepsilon} + t \right| \right] \Big|_{\varepsilon = h_\theta^{-1}(t, \mathbf{z}_t)}. \tag{40}$$

E.2. Objective function

To train a model with such a specific reverse process, we can utilize a slightly modified NDMs framework. The only component of the NDMs' objective that is unclear is the diffusion term $\mathcal{L}_{\text{diff}}$. NDMs provide a conditional reverse SDE



(a) DDPM with regular reverse process.

(b) NDM with restricted (OT) reverse process.

Figure 3. Comparison of DDPM and NDM with restricted reverse process to be optimal transport, 1D distribution.

associated with the forward process (28) in the following form:

$$d\mathbf{z}_t = f^f_\varphi(x, t, \mathbf{z}_t)dt + g(t)d\bar{\mathbf{w}}. \tag{41}$$

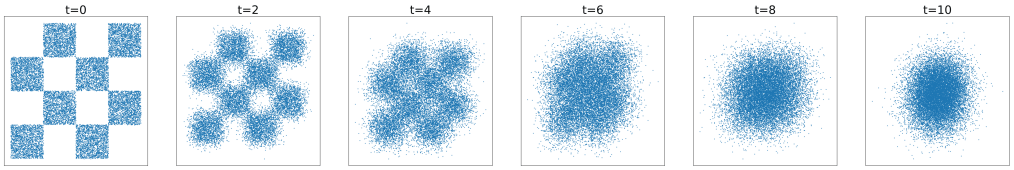
Also, here we have the reverse SDE (38). Therefore, we may find diffusion term $\mathcal{L}_{\text{diff}}$ of objective as follows:

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{q(\mathbf{x})}\mathbb{E}_{u(t)}\mathbb{E}_{q(\mathbf{z}_t|\mathbf{x})}\frac{1}{g^2(t)}\left\|f^f_\varphi(x, t, \mathbf{z}_t) - f^r_\theta(t, \mathbf{z}_t)\right\|^2_2. \tag{42}$$

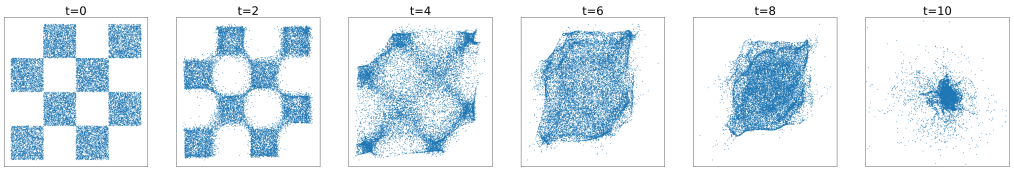
E.3. Results and discussion

Figures 3a and 3b illustrate trajectories learned by DDPM and NDM with learnable F_φ and restricted reverse process. As expected, DDPM learns curved trajectories predetermined by fixed forward process. At the same time NDM effectively learns dynamic OT. It worths noting that DDPM with the restricted reverse process is by design not able to learn the data distribution, since it's impossible to match the fixed forward process (with curved trajectories) with the reverse process (with straight trajectories).

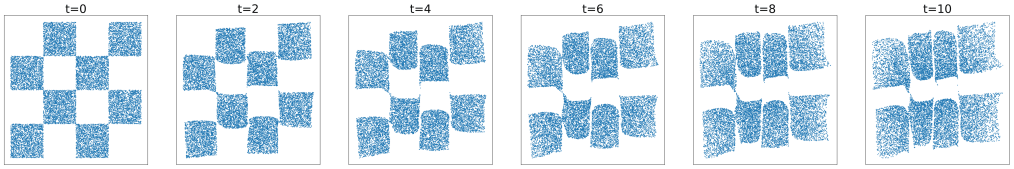
The proposed approach is limited to 1D data, monotonically increasing $\hat{\mathbf{x}}_\theta$, and a nontrivial h^{-1}_θ function, which we resolve using 5 iterations of Newton's method. Nevertheless, this experiment clearly demonstrates that NDMs may be utilised for learning OT as well as other (e.g. computationally efficient ones) dynamics by restricting the reverse process. Establishing rigorous theoretical connections with OT, developing specific techniques for efficient parameterisation of the reverse process and generalising to higher dimensions are interesting avenue for future work.



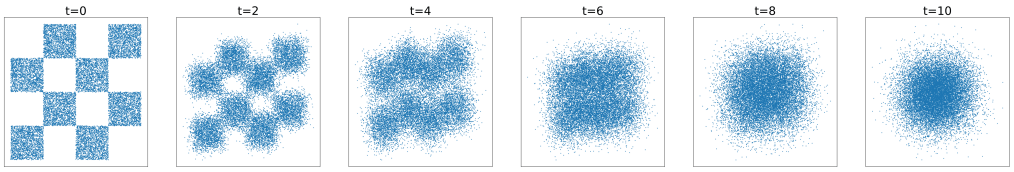
(a) DDPM, samples $\mathbf{z}_t$ from forward process.



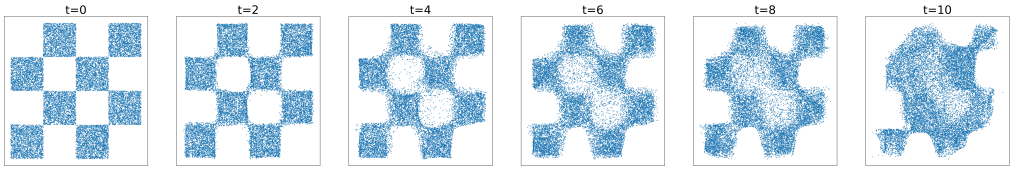
(b) DDPM, predictions $\hat{\mathbf{x}}_\theta(\mathbf{z}_t, t)$ for different time steps.



(c) NDM, forward transformations $F_\varphi(\mathbf{x}, t)$.

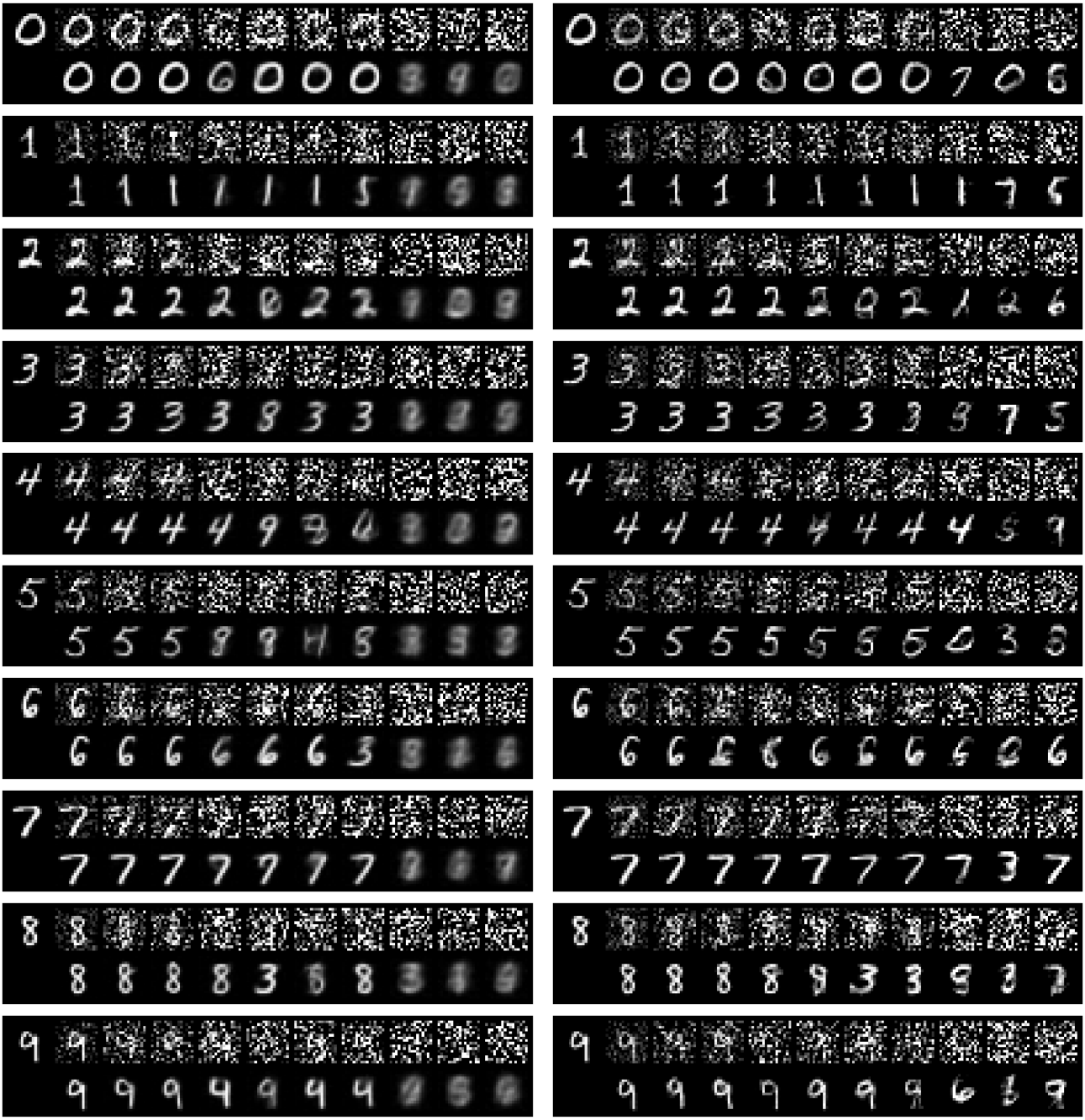


(d) NDM, samples $\mathbf{z}_t$ from forward process.



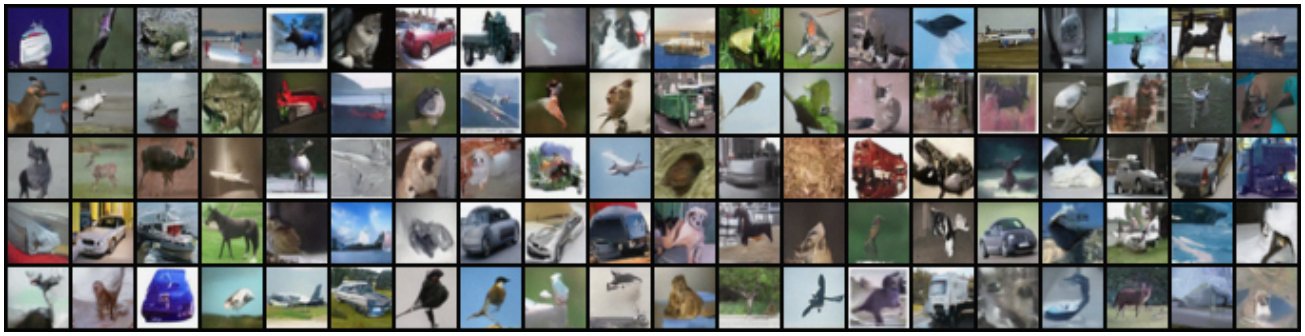
(e) NDM, predictions $\hat{\mathbf{x}}_\theta(\mathbf{z}_t, t)$ for different time steps.

Figure 4. Comparison of DDPM and NDM on 2D distribution.

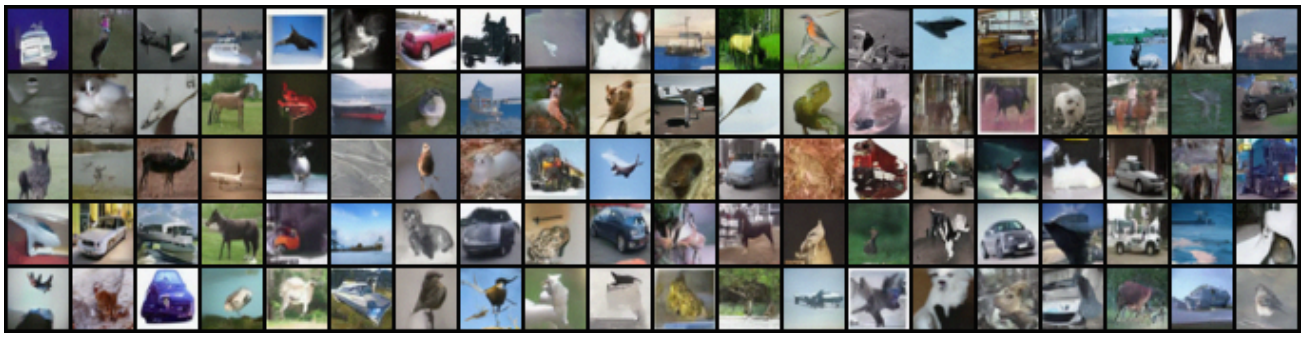


(a) DDPM (b) NDM

Figure 5. Samples $\mathbf{z}_t$ from forward process and predicted data points $\hat{\mathbf{x}}_\theta(\mathbf{z}_t, t)$ on MNIST. (a) Samples from DDPM. (b) Samples from NDM. In each group, *Left:* data sample, *Top:* noised samples $\mathbf{z}_t$, *Bottom:* predicted data points $\hat{\mathbf{x}}_\theta(\mathbf{z}_t, t)$.



(a) DDPM, FID = 11.44



(b) NDM, FID = 11.82

Figure 6. Samples on CIFAR-10. (a) Samples from DDPM. (b) Samples from NDM. Samples of both models are generated with the same random seed.