Residual Learning in Diffusion Models

Junyu Zhang[1] Daochang Liu[2] Eunbyung Park[3] Shichao Zhang[4] Chang Xu[2*]
[1]Central South University, [2]The University of Sydney
[3]Sungkyunkwan University, [4]Guangxi Normal University
zhangjunyu@csu.edu.cn, {daochang.liu, c.xu}@sydney.edu.au,
epark@skku.ac.kr, zhangsc@gxnu.edu.cn

Abstract

Diffusion models (DMs) have achieved remarkable generative performance, particularly with the introduction of stochastic differential equations (SDEs). Nevertheless, a gap emerges in the model sampling trajectory constructed by reverse-SDE due to the accumulation of score estimation and discretization errors. This gap results in a residual in the generated images, adversely impacting the image quality. To remedy this, we propose a novel residual learning framework built upon a correction function. The optimized function enables to improve image quality via rectifying the sampling trajectory effectively. Importantly, our framework exhibits transferable residual correction ability, i.e., a correction function optimized for one pre-trained DM can also enhance the sampling trajectory constructed by other different DMs on the same dataset. Experimental results on four widely-used datasets demonstrate the effectiveness and transferable capability of our framework.

1. Introduction

Diffusion models (DMs) [24, 64], a recent family of generative models, are revolutionizing the field of image generation [13, 16, 54, 56], and also finding rapid applications in other domains such as video generation [20, 48], 3D generation [61, 76], text-to-image generation [51, 57, 58], inverse problems [11, 41, 68], and image segmentation [3, 40]. Especially since the emergence of the seminal work [69], DMs are generalized through the lens of stochastic differential equations (SDEs), achieving start-of-the-art image quality [30, 54] and good mode coverage [32, 34, 45, 67].

DMs consist of two processes, *i.e.*, forward diffusion and reverse sampling [62, 65, 72]. In the diffusion process, a forward SDE is employed to perturb the data distribution via a well-designed noise schedule [28, 69]. Since the noise scale gradually reaches its maximum value, the target data distribution reverts to a prior distribution. Sample generation can be achieved by reversing this process [29, 46, 78]. Crucially, the reverse process satisfies a reverse-time SDE or a probability flow (PF) ordinary differential equation (ODE) [69]. This can be derived from the forward SDE by considering the score of the marginal probability densities as a function of time [1, 47]. We can therefore approximate the reverse-time S/ODE by training a time-dependent neural network to estimate the score [26, 64, 66].

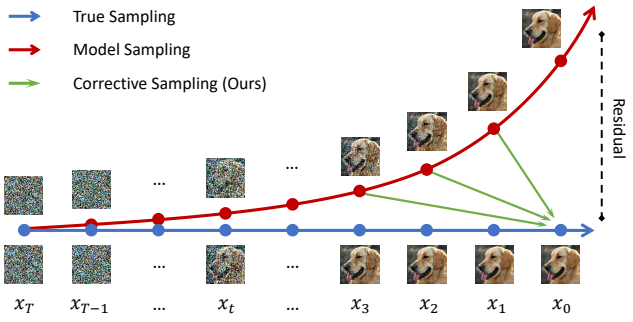

Figure 1. **Residual Learning in Diffusion Models.** A gap arises between true sampling and model sampling due to score estimation and discretization errors. Our residual learning framework employs a parameterized correction function to learn the residual and correct the model sampling towards true sampling process.

In light of SDE frameworks, DMs tend to be based as directly as possible on a rigorous theoretical footing [28, 47], which serves as a solid evident that a gap between the true and modeling distributions constructed by the model sampling trajectory cannot be avoided [29, 30, 32, 78], as shown in Figure 1. Though the approximated reverse-time S/ODE enables to generate images, the modeling distribution constructed by them cannot completely match the true distribution [30, 45]. Because the model sampling trajectory gradually deviates from the true sampling trajectory, leading to an increasingly widening gap. Hence, the gap leads to a residual error in generated images that reduces image fidelity.

The primary reason is that two errors gradually accumu-

*Corresponding author

late throughout the model sampling process [7, 32], which are score estimation error [19, 32] and discretization error [29, 78]. The former one comes from the estimation of the expectation of the score at each noise scale [2, 15]. In practice, it is intractable to solve the exact solution of score since we can not access all the samples, and the solution is approximated instead via the law of large numbers [24, 64], leading to a score estimation error. As for the latter, the error source is in that we cannot directly solve the approximated reverse-time S/ODE due to the high dimensionality of the data [26]. Instead, the reverse-time S/ODE needs to be discretized as a sampling trajectory that consists of finite time steps. An approximate solution can be obtained by using numerical solvers to iteratively solve the integral between two adjacent time steps [28, 56]. However, the discrete sampling strategy brings discretization error in each time step and accumulates all the error during sampling, especially when using a small number of steps with large integral interval in the fast sampling field [46, 78, 79].

Motivated by these observations, we propose a novel residual learning framework that employs a correction function to rectify the model sampling trajectory, thereby reducing the sampling gap. Our core idea is to optimize a parameterized correction function to guide the sampling trajectory to the true data distribution, as shown in Figure 1. Specifically, we disassemble the generative process into two phases which combines an approximated reverse-time S/ODE [69] with our optimized correction function. In the first phase, the approximated reverse-time S/ODE is used to generate samples from prior distribution to an intermediate time step in the sampling trajectory that has not deviated from the true data distribution too far. In the subsequent phase, our optimized correction function guides those samples towards the true data distribution more closely.

In this manner, our residual learning framework enhances the generative performance of pre-trained DMs without the need to fine-tune or train them from scratch. Additionally, our correction function, trained for a certain DM, can improve arbitrary sampling trajectories of different DMs on the same dataset, showcasing a significant transferable capability. Once the dataset is specified, the matched score becomes the sole variable in DMs-based image generation, exhibiting slight variations due to the capacity differences among different DMs. Consequently, the sampling trajectories modeled by different DMs are similar, given that the prior and true distributions are fixed, resulting in similar sampling error patterns. In this context, our proposed residual learning captures this shared error pattern and possesses transferable correction capability.

Our contributions can be summarized as follows. 1) We analyse the gap between true distribution and modeling distribution due to the score estimation and discretization errors. 2) We propose a novel residual learning framework employing a correction function to rectify the sampling trajectory, as a plug-and-play improvement on pre-trained diffusion models. 3) We verify the effectiveness of our framework on various benchmark datasets and demonstrate its transferable capability across different DMs.

2. Related Works

Diffusion Models Diffusion probabilistic models [13, 24, 34, 50] and score-based models [64, 69] are families of the generative models that generate samples from a prior distribution by using reverse-S/ODE integrators [27, 28], and a certain parameterization reveals an equivalence between them [47]. Compared to previous generative models, such as VAEs [35], flow-based models [14], and GANs [4, 17], DMs generate samples with start-of-the-art image quality and comparable likelihood [32, 45, 67]. One line of works resolve the slow sampling of diffusion models and achieve significant acceleration in the generation process, such as DDIM [63], PNDM [42], DPM-solver [46], DEIS [78], and SciRE-Solver [39]. However, they exacerbate the discretization error due to the larger integral interval, resulting in a reduction in image quality.

Sampling Gap Recently, some methods make great efforts to reduce sampling gap via using high order numerical solvers [39, 78, 79, 81]. Stable diffusion [56] matches the score in latent space [72], which naturally reduce the discretization error by solving the integral in lower dimension. DMCMC [29] utilizes MCMC to obtain a good initialization points close to the modeling distribution. Besides, [6, 30] propose to adjust the matched score via a robust discriminator. Contrary to the usual treatment, certain works [33, 49, 59, 70] mitigate the sampling gap in small time steps of model sampling by distilling knowledge from larger sampling steps. Orthogonal to them, we introduce a residual learning framework to narrow the sampling gap by using an optimized correction function to rectify the sampling process of pre-trained DMs.

Transfer Learning Large-scale models acquire valuable feature representations that demonstrate effective transferability [5, 8, 55] to unseen tasks, datasets, and domains [25, 38, 71, 77]. In generative modeling, recent works propose transferring the weights of pre-trained models from a source domain (e.g., faces) to a new domain (e.g., portraits of one person) [18, 36, 52, 53]. Different from them, we transfer the knowledge of the residual to rectify the sampling trajectory of arbitrary diffusion model on the same data set.

3. Preliminary

3.1. Sampling Gap Analysis

DMs [24, 64] are a class of generative models, synthesizing images by gradually denoising random points sampled from prior distribution. Specifically, for a given D-dimensional

data $x_0 \sim p_{data}(x_0)$ sampled from the real distribution, one can utilize a forward SDE to perturb them

$$dx = \boldsymbol{F}_t x dt + \boldsymbol{G}_t d\omega, \tag{1}$$

where $\boldsymbol{F}_t \in \mathbb{R}^{D\times D}$ denotes the linear drift coefficient, $\boldsymbol{G}_t \in \mathbb{R}^{D\times D}$ denotes the diffusion coefficient, ω is a standard Wiener process and $t \sim U[0,1]$. Under some mild assumptions [69], the forward SDE in Eq. (1) is associated with a reverse-time diffusion process

$$dx = \left[\boldsymbol{F}_t x - \boldsymbol{G}_t \boldsymbol{G}_t^T \nabla \log p_t(x)\right] dt + \boldsymbol{G}_t d\omega, \tag{2}$$

where ω denotes a standard Wiener process in the reverse-time direction, and $\nabla \log p_t(x)$ is the gradient of the log probability density with respect to the perturbed data at time step t, a.k.a. score [26, 74]. In theory, with a known prior distribution π, such as the normal distribution, one can generate samples via solving Eq. (2) [1].

Discretization Error However, it is intractable to directly solve the integral in Eq. (2). Alternatively, it is split into T discretization steps $[t_1 = 0, \cdots, t_t, \cdots, t_N = T]$ with $T-1$ intervals, and sum each interval between time step $t+1$ to t equals to the integral from time step T to 0

$$x_t = x_{t+1} + \frac{1}{2}\int_{t+1}^{t} \left[\boldsymbol{G}_\tau \boldsymbol{G}_\tau^T \nabla \log p_\tau(x)\right] d\tau + \boldsymbol{G}_t z_t, \tag{3}$$

where $z_t \sim \mathcal{N}(0, I^D)$. In practice, one can utilize a numerical solver [28, 39, 46, 78] to solve each integral interval

$$x_t = x_{t+1} + \frac{\Delta t}{2} \boldsymbol{G}_{t+1} \boldsymbol{G}_{t+1}^T \nabla \log p_{t+1}(x) + \boldsymbol{G}_t z_t. \tag{4}$$

Here, Δt is the integral interval between time step $t+1$ to t. Obviously, using Eq. (4) to solve each integral instead of Eq. (3) will cause the discretization error, and will accumulate them to a large error.

Score Estimation Error On the other hand, $\nabla \log p_t(x)$ is inaccessible due to the high dimensionality of data, which leads to the analytical intractability of the probability density function [26]. To remedy this, prior works [64, 66, 74] design the loss function $\mathcal{J}_{\mathrm{SM}}$ to match the score $\nabla \log p_t(x)$, wherein a time-dependent network $s_\theta(x_t, t)$ is employed to approximate it

$$\mathcal{J}_{\mathrm{SM}}(\theta;\omega) = \frac{1}{2}\int_0^1 \mathbb{E}_{x_0,x_t}\left[\omega(t)\left\|\nabla \log p_{0t}(x_t|x_0) - s_\theta(x_t,t)\right\|_2^2\right] dt.$$

Here, $\nabla \log p_{0t}(x_t|x_0)$ has a closed form expression as $p_{0t}(x_t|x_0)$ is a simple Gaussian distribution obtained from a given SDE [69], and $\omega(t)$ denotes a time-dependent weighting function used for stable training.

In practice, $\mathcal{J}_{\mathrm{SM}}(\theta;\omega)$ is optimized using empirical samples via Monte Carlo methods [26, 64, 65], thus leading to the score estimation error. Because the exact solution of $\nabla \log p_{0t}(x_t|x_0)$ requires each sample of the corresponding distribution $p_{0t}(x_t|x_0)$, which is impossible. In addition, previous models [28, 32, 45, 67, 70] have the trade-off between model likelihood [45, 67] and image quality [30, 54] via setting the smallest time step ϵ

$$\frac{1}{2}\int_\epsilon^1 \mathbb{E}_{x_0,x_t}\left[\omega(t)\left\|\nabla \log p_{0t}(x_t|x_0) - s_\theta(x_t,t)\right\|_2^2\right] dt. \tag{5}$$

Here, score beneath ϵ will not be estimated, further aggravating the score estimation error when using the unmatched score for sampling.

Sampling Gap Once the score network $s_\theta(x_t,t) \approx \nabla \log p_t(x)$ is matched for almost all $x \in \mathbb{R}^D$ and $t \sim U[\epsilon, 1]$, it can be used to generate new samples by solving Eq. (4) with $\nabla \log p_{t+1}(x)$ replaced by $s_\theta(x_{t+1}, t+1)$

$$x_t = x_{t+1} + \frac{\Delta t}{2} \boldsymbol{G}_{t+1} \boldsymbol{G}_{t+1}^T s_\theta(x_{t+1}, t+1) + \boldsymbol{G}_t z_t. \tag{6}$$

However, score estimation and discretization errors manifest simultaneously and gradually accumulate throughout the model sampling trajectory, leading to a sampling gap between the true distribution and the model distribution.

3.2. Residual Definition

Building upon the previous analysis, the sampling gap introduces a residual in the generated images, resulting in a reduction in image quality. Below we provide a definition of this residual. For any given forward SDE, its corresponding deterministic sampling process can be formulated as

$$\frac{dx}{dt} = \boldsymbol{F}_t x - \frac{1}{2}\boldsymbol{G}_t \boldsymbol{G}_t^T \nabla \log p_t(x), \tag{7}$$

which is the probability flow (PF) ordinary differential equation (ODE) [69, 70]. Concretely, arbitrary x_t in the PF ODE trajectory can revert to $x_0 \sim p_{data}(x_0)$ via Eq. (7). However, we can not obtain x_0 via directly solving this PF ODE, and instead, we generate x_0^{ODE} via using $s_\theta(x_{t+1}^{\mathrm{ODE}}, t+1)$ to replace $\nabla \log p_{t+1}(x)$ and using the iterative rule below

$$x_t^{\mathrm{ODE}} = x_{t+1}^{\mathrm{ODE}} + \frac{\Delta t}{2} \boldsymbol{G}_{t+1} \boldsymbol{G}_{t+1}^T s_\theta(x_{t+1}^{\mathrm{ODE}}, t+1). \tag{8}$$

For a given x_T, x_0^{ODE} can not completely match x_0 because of the following residual

$$\Omega = x_0 - x_0^{\mathrm{ODE}}. \tag{9}$$

4. Residual Learning Framework

We put forward a novel residual learning framework, via employing a parameterized correction function to rectify

the sampling trajectory modeled by a pre-trained DM $s_{\theta^*}(x_t, t)$, as shown in Figure 2. Moreover, we intuitively and theoretically analyze the transferable correction capability of our framework. Concretely, a correction function optimized for a given pre-trained DM can correct arbitrary sampling trajectory of different DMs trained on the same data. Below we introduce the residual learning and corrective sampling of our framework, provide an analysis on its transferable capability, plus a discussion on the optimization and the efficient correction.

4.1. Overview

In practice, it is challenging to collect the residual pair $\left(x_0, x_0^{\mathrm{ODE}}\right)$. While we define the residual in Eq. (9), it is intractable obtaining Ω by finding a true sample x_0 paired with a given generated sample x_0^{ODE} via directly solving Eq. (7). However, conversely, it is possible to simulate the generated sample x_0^{ODE} with a given true image x_0 from the training dataset. Heuristically, we can first obtain x_t by adding noise to the true image x_0 and reverse it to x_0^{ODE} with Eq. (8). Under such a perspective, we are able to construct the residual pair $\left(x_0, x_0^{\mathrm{ODE}}\right)$ for a simulation of the residual. Thereupon, we design a two-stage residual learning framework to learn the residual as in Algorithm 1.

Residual Learning In the first stage, we utilize a given forward SDE to perturb the data x_0 and obtain x_t which reside on the true sampling trajectory $\{x_t\}_{t\in[0,T]}$. For brevity, we use VE SDE [69] to describe our framework

$$x_t = x_0 + \sqrt{\sigma^2(t) - \sigma^2(0)}z_t, \tag{10}$$

where $\sigma(t)$ is a designed noise schedule [24, 28, 69], and z_t is sampled from the Gaussian distribution. Based on previous analysis, the perturbed x_t can reverse to x_0^{ODE} when replacing $\nabla \log p(x_t, t)$ with an optimized score function $s_{\theta^*}(x_t^{\mathrm{ODE}}, t)$ and using the iterative rule to solve Eq. (8). In this context, we successfully simulate the residual via constructing the residual pair $\left(x_0, x_0^{\mathrm{ODE}}\right)$.

In the second stage, to learn the residual, we define the correction function as

$$\boldsymbol{f} : (x_i^{\mathrm{ODE}}, i) \mapsto x_0, \tag{11}$$

where $i \in (\epsilon, t)$, t is defined in Eq. (10). To avoid unmatched score in $[0, \epsilon]$ mentioned in Eq. (5), we use x_i^{ODE} instead of x_0^{ODE} to construct the residual pair. On the other hand, in practice, it is also difficult to indeed reverse to x_0^{ODE}. We thus utilize x_i^{ODE} to learn the residual. To define i, we first discretize the time horizon $[\epsilon, T]$ into $N-1$ subintervals, with boundaries $n_1 = \epsilon < n_2 < \cdots < n_{t-1} < n_t < \cdots < n_N = T$, where i belongs to $[n_2, \cdots, n_{t-1}]$.

Corrective Sampling For an optimized correction function $\boldsymbol{f}_{\phi^*}(\cdot, \cdot)$ trained on a given dataset, we can combine it with reverse-time S/ODE approximated by $s_{\theta^*}(x_t, t)$ of arbitrary DM trained on the same dataset. Concretely, we first sample a batch of images x_T from the prior distribution and use a corresponding reverse-time S/ODE to reverse

Algorithm 1: Residual Learning

```
def residual_learning(net, sigma, x_0, t_steps
    ):
  """
  net is the pre-trained DM
  sigma is the noise schedule
  x_0 are a batch of training images
  t_steps is ODE reverse schedule
  e.g. [n_2, ... n_t-1]
  """

  #add noise via VE SDE
  t = randint(0, N)
  noise = sqrt(sigama(t)**2 - sigama(0)**2)
  x_t = x_0 + noise

  #the last step of ODE reverse
  i = random.choice(t_steps)

  #use PF ODE to reverse samples
  for j in range(t, i - 1, -1):
    x_t = ODE_solver(net, x_t, t_steps[j])

  #the last x_t is x_i
  #residual prediction using the correction
    function
  f_x_i = c_function(x_t)

  #loss backpropogation
  loss = d(f_x_i, x_0)
  loss.backward()
```

Algorithm 2: Corrective Sampling

```
def corrective_sampling(net, c_function,
    epsilon, x_t, t_steps):
  """
  net is the pre-trained DM
  c_function is a correction function
  epsilon is the smallest step in Eq.(3)
  x_t sampled from prior distribution
  t_steps is sampling time schedule
  e.g. [80, ... , 0.58, ... , 0.002]
  """

  #adaptive correction strategy
  c = max(0.01, epsilon)

  for t in t_steps:
    #use reverse O/SDE to generate samples
    x_t = O/SDE_solver(net, x_t, t)

    #use our correction function to correct
    if t <= c:
        x_t = c_function(x_t)
        break

  return x_t
```

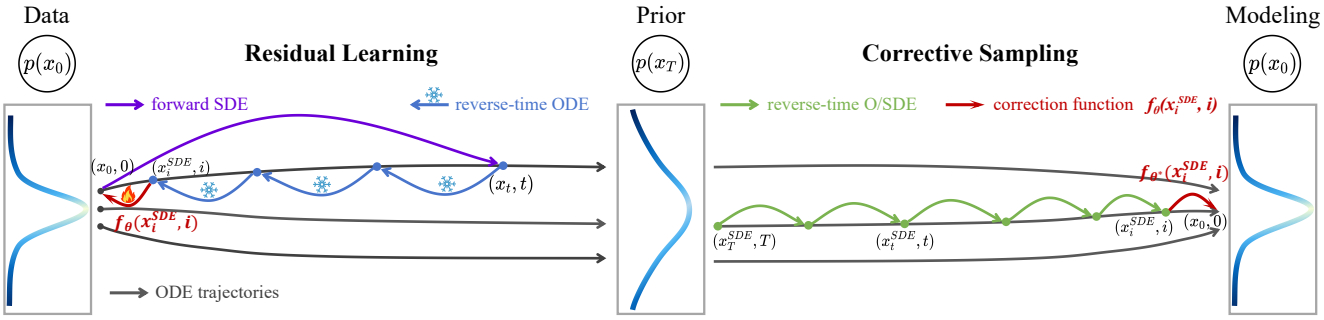

Figure 2. The overall structure of our Residual Learning Framework. For residual learning, we first diffuse x_0 to x_t with a forward SDE and utilize a corresponding reverse-time ODE to reverse it back to x_i^{SDE}, and train a parameterized correction function for mapping x_i^{SDE} to x_0. For sampling, a reverse-time S/ODE iterative diffuse x_T to x_i^{SDE} and an optimized correction function maps x_i^{SDE} to x_0 accordingly.

them to $x_{i^*}^{\text{SDE}}$ or $x_{i^*}^{\text{ODE}}$. For brevity, we utilize the notation $x_{i^*}^{\text{SDE}}$ here. Subsequently, we utilize the optimized correction function to guide $x_{i^*}^{\text{SDE}}$ toward x_0, where i^* represents the time step to apply $\boldsymbol{f}_{\phi^*}(\cdot,\cdot)$, which is a hyper-parameter detailed in Algorithm 2.

For the choice of the hyper-parameter i^*, we design an adaptive strategy to best make the capability of our framework. As mentioned earlier, samples near a small ϵ are scarce, which leads to imprecise score matching [32]. Owing to the properties of our framework, a large i^* will not influence the performance of our correction function. Hence, we enable to set a relatively large i^* to bypass the unmatched score. On the other hand, ϵ defined in Eq. (4) varies greatly, fo r instance, one can choose $\epsilon = 10^{-2}$ or $\epsilon = 10^{-5}$ [67]. Therefore, we establish a unified standard for stable correction with ϵ as reference. Concretely, we set i^* as the nearest and larger step to c and keep this setting across all tasks and datasets, where $c = \max\{0.01, \epsilon\}$. In this manner, our framework can correct the sampling trajectory, leading to a reduction on the residual and an improved image quality.

4.2. Transferable Residual Correction

Adhering to the design philosophy of our framework, the correction function has the property: its outputs are consistent for arbitrary pairs of (x_i^{ODE}, i) that belong to the same PF ODE trajectory, *i.e*, $\boldsymbol{f}(x_a^{\text{ODE}}, a) = \boldsymbol{f}(x_b^{\text{ODE}}, b)$ for all $a, b \in (0, t]$. Hence, the optimized correction function can guide samples near $x_{i_{\in(0,t)}}^{\text{ODE}}$ to true distribution, for instance, $x_{i_{\in(0,t)}}^{\text{SDE}}$. On the other hand, for a given D-dimensional data $x_0 \sim p_{data}(x_0)$, different DMs all aim to model a distribution which overlaps with the true distribution. The PF ODE sampling trajectory for a given dataset is fixed since the score $\nabla \log p_t(x)$ is invariable, and the corresponding SDE sampling trajectory always near to the ODE trajectory. Hence, the sampling trajectories modeled by different DMs are similar, given that the prior and true distributions are fixed, resulting in the same sampling error pattern. In this sense, samples x_i^{SDE} in the same time step i that are modeled by different DMs fall within the similar domain, thus can be rectified by the same optimized correction function.

Motivated by this understanding, for a given dataset, we can use our optimized correction function to correct any sampling trajectory modeled by different DMs, dubbed transferable residual correction ability. Below we provide a theoretical justification for the transferable capability of our framework based on asymptotic analysis.

Theorem 1 *Let* $\Delta i := \max_{(i,i_\xi)\in[\epsilon,t]}\{|i - i_\xi|\}$. *Assume* $\boldsymbol{f}_\phi(\cdot,\cdot)$ *satisfies the Lipschitz condition: there exists* $L > 0$ *such that for all* (x, i) *and* (y, i_ξ), *we have* $\|\boldsymbol{f}_\phi(\mathrm{x}, i) - \boldsymbol{f}_\phi(\mathrm{y}, i_\xi)\|_2 \le L\|\mathrm{x} - \mathrm{y}\|_2$. *Assume further that the correction function called at* i *has local error uniformly bounded by* $O\left((i - i_\xi)^{p+1}\right)$ *with* $p \ge 1$. *Then if* $\mathcal{L}(\phi; \lambda, \beta) = 0$, *we have*

$$\sup_{\mathrm{x},\mathrm{y}} \|f_\phi(\mathrm{x}, i) - f_\phi(\mathrm{y}, i_\xi)\|_2 = O\left((\Delta i)^p\right).$$

Proof. For a fixed PF ODE trajectory corresponds to a given dataset, any modeling sampling trajectory is near to the PF ODE trajectory. Hence, (x, i) is close to (y, i_ξ) since i is close to i_ξ, and our correction function can naturally map them to true distribution.

Based on Theorem 1, our correction function, optimized on a fixed PF ODE sampling trajectory for a given pre-trained DM, can correct any model sampling trajectory of different DMs trained on the same dataset. We will verify the transferable capability of correction function in our experiment.

4.3. Optimization

To learn the residual and enhance the transferable correction capability, below we delve into the optimization of our correction function. For a given correction function $\boldsymbol{f}(\cdot,\cdot)$, we have $\boldsymbol{f}(x_i^{\text{ODE}}, i) = x_0$, where $i \in (0, t]$. Suppose we have a free-form deep neural network to represent our correction

function, we can train it with our correction loss

$$\mathcal{L}(\phi;\lambda)=\mathbb{E}\left[\lambda(i)d(\boldsymbol{f}_\phi(x_i^{\mathrm{ODE}},i),x_0)\right],$$

where $\lambda(\cdot)$ is a positive time-dependent weighting function, and $d(\cdot,\cdot)$ is a metric function that satisfies $\forall \mathrm{x},\mathrm{y}: d(\mathrm{x},\mathrm{y})\geq 0$ and $d(\mathrm{x},\mathrm{y})=0$ if and only if $\mathrm{x}=\mathrm{y}$. In our experiments, we consider the squared l_2 norm $d(\mathrm{x},\mathrm{y})=\|\mathrm{x}-\mathrm{y}\|_2^2$, l_1 norm $d(\mathrm{x},\mathrm{y})=\|\mathrm{x}-\mathrm{y}\|_1$, and the Learned Perceptual Image Patch Similarity (LPIPS) [70, 80].

To further improve the transferable ability of our framework, we utilize some widely used data augmentation techniques [8, 21, 22, 43] to reformulate our correction loss

$$\mathcal{L}(\phi;\lambda,\beta)=\mathbb{E}\left[\lambda(i)d(\boldsymbol{f}_\phi(x_i^{\mathrm{ODE}},i),x_0)\right]+\mathbb{E}\left[\beta d(\boldsymbol{f}_\phi(x_0^+),x_0)\right],$$

where x_0^+ represents the corresponding image augmented from x_0, β is a positive weight which performs well across all tasks and datasets when $\beta=1$. In practice, we utilize several data augmentation methods to obtain (x_0^+), for instance, add small order of magnitude Gaussian noise to x_0, rotate and crop x_0, as well as mask some pixels [22].

For training our parameterized correction function $\boldsymbol{f}_\phi(\cdot,\cdot)$, we minimize the $\mathcal{L}(\phi;\lambda,\beta)$ by stochastic gradient descent on the model parameters ϕ, with detailed information given in Appendix.

4.4. Efficient Residual Correction

To avoid worsening the issue of slow sampling in DMs, we propose an efficient residual correction method, where the correction function is parameterized by a lightweight backbone. Concretely, we distill correction ability from an optimized correction function $\boldsymbol{f}_{\phi^*}(\cdot,\cdot)$ parameterized by heavy parameters ϕ^* to the one with light parameters ϕ^-

$$\mathcal{L}(\phi^-;\alpha)=\mathbb{E}\left[d(\boldsymbol{f}_{\phi^*}(x_i^{\mathrm{ODE}},i),\boldsymbol{f}_{\phi^-}(x_i^{\mathrm{ODE}},i))\right].$$

Similarly, we train the lightweight correction function through stochastic gradient descent on the model parameters ϕ^-. Once optimized, the lightweight correction functions $\boldsymbol{f}\phi^-(\cdot,\cdot)$ are $4\sim 64$ times smaller than their heavier versions $\boldsymbol{f}\phi^*(\cdot,\cdot)$, with only a slight drop in image quality.

5. Experiments

5.1. Setting and Implementation

To evaluate the performance of our residual learning framework, we conduct groups of experiments on four benchmark datasets, including CIFAR-10 [37], CelebA 64×64 [44], AFHQv2 64×64 [10], ImageNet 256×256 and 512×512 [73]. In our main experiments, our correction function is combined with pre-trained DMs to improve them. For CIFAR-10, we employ NSCNv2 [65], DDPM [24], SDE series [69], and EDM [28] as pre-trained DMs. Our correction functions are implemented with the architecture of the corresponding pre-trained DMs. For ImageNet, we utilize ADM [13], LDM [56], and DiT [54] as pre-trained DMs, and the correction function is implemented with the architecture of ADM. For CelebA and AFHQv2, we employ EDM [28], FDM [75], DDPM++ [69], STDDPM [32], Soft Diffusion [12], INDM [31] as pre-trained DMs, and choose the architecture of EDM to formulate the correction function. The correction function is randomly initialized.

During training, we exclude the time embedding setting from the architecture of DM and use it to construct correc-

Pre-trained Model	FID↓	IS↑
ImageNet 256×256 (class-conditional)		
ADM [13]	10.94→9.86	100.98→111.01
ADM-G [13]	4.59→4.47	186.70→191.23
LDM-4 [56]	10.56→10.35	103.49→114.62
LDM-8 [56]	15.51→14.47	79.03→84.68
DiT [54]	9.62→9.49	121.50→127.84
ImageNet 512×512 (class-conditional)		
ADM [13]	23.24→19.97	58.06→61.54
ADM-G [13]	7.72→7.65	172.71→180.62
DiT [54]	12.03→11.37	105.25→121.53

Table 1. Performance on ImageNet. Models in the first column represents the baseline pre-trained DMs. The values before → are the results of baseline pre-trained DMs on ImageNet, and values after → are results improved by our residual learning framework.

Pre-trained Model	FID↓	IS↑
NSCNv2 [65]	10.87→9.89	8.40→8.73
DDPM [24]	3.17→3.05	9.46→9.50
SDE (VE) [69]	2.55→ 2.44	9.83→9.86
SDE (deep, VE) [69]	2.20→ 2.15	9.89→9.92
EDM [28]	2.04→1.93	9.84→9.89

Table 2. Performance on CIFAR-10. Models in the first column represents the baseline pre-trained DMs. Analogously, values before → are the results obtained from baseline pre-trained DMs, and values after → are results improved by our framework.

AFHQv2 64×64		CelebA 64×64	
Pre-trained Model	FID↓	Pre-trained Model	FID↓
EDM VE [28]	2.16→2.10	DDPM++ [69]	2.32→2.22
EDM VP [28]	1.96→1.91	STDDPM [32]	1.90→1.85
FDM VP [75]	2.39→2.30	Soft Diffusion [12]	1.85→1.80
FDM VE [75]	47.30→ 42.10	INDM [31]	1.75→1.71

Table 3. Performance on AFHQv2 and CelebA. The values before → are the results obtained from baseline pre-trained DMs, and values after → are results improved by our framework.

Figure 3. Randomly selected 512×512 images improved by our residual learning framework.

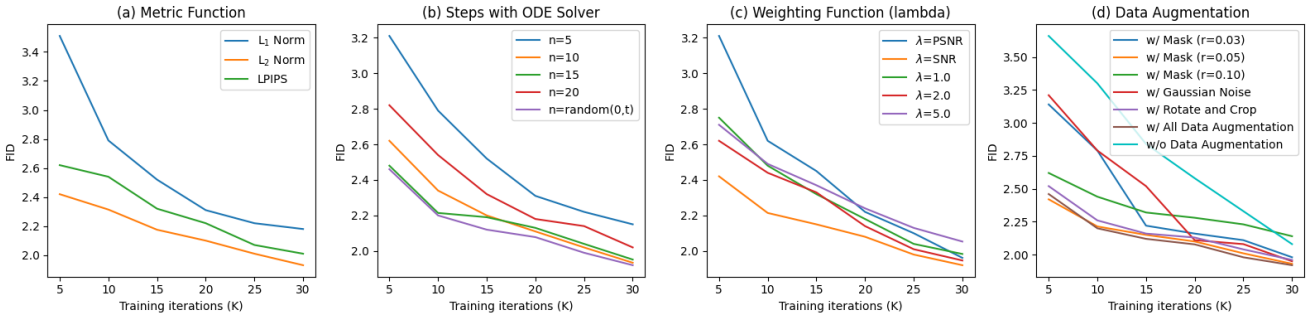

Figure 4. Ablation study on different optimization strategies evaluated on CIFAR-10. (a) we train the correction function with different metric functions, wherein L_2 norm performs better. (b) we use PF ODE to reverse x_t to x_i with n steps, and random choice of n achieves the best performance. (c) we use different weighting functions to optimize our correction loss, such as signal-to-noise ratio (SNR) [34], peak signal-to-noise ratio (PSNR) [9], and also constant values. In practice, we find SNR weighting schedule performs best. (d) we validate that combining all data augmentation methods, including mask pixel, add Gaussian noise, rotate and crop, leads to the best correction ability.

Correction Function	FID↓	IS↑
EDM-H $\equiv [256, 204\text{M}]$	2.04→1.93	9.84→9.91
EDM-XL $\equiv [128, 51\text{M}]$	2.04→1.93	9.84→9.89
EDM-L $\equiv [96, 28.7\text{M}]$	2.04→1.94	9.84→9.89
EDM-M $\equiv [64, 12.7\text{M}]$	2.04→1.95	9.84→9.86
EDM-S $\equiv [32, 3.2\text{M}]$	2.04→1.99	9.84→9.84
EDM-T $\equiv [16, 0.8\text{M}]$	2.04→2.03	9.84→9.80

Table 4. Ablation study on the size of correction function. $[\cdot, \cdot]$ represents the pair of model channels and model size. For instance, EDM-H$\equiv$ $[256, 204\text{M}]$ is the correction function constructed by EDM backbone with 256 channels and the resulting model size is 204M [28]. Values before $\rightarrow$ are the results obtained from baseline pre-trained DMs, and values after $\rightarrow$ are results improved by our framework. Obviously, the performance of correction function shows a gradual improvement with an increase in the model size, but this improvement does not continue indefinitely.

tion function. We retain only the input as a batch of images, and the output is also images, with a slightly reduced model size. Besides, we augment the training images with data augmentation approaches that mentioned in section 4.3.

5.2. Results

To quantitative compare the correction performance, we utilize standard metrics, including Fréchet Inception Distance (FID) [23] and Inception Score (IS) [60], to verify them on 50,000 generated samples. In the experimental results on CIFAR-10, our framework enhances the generation performance, indicating that correction function effectively learns the residuals and, consequently, enhances the overall image quality, seen in Table 2. Our framework also demonstrate the effectiveness on CelebA and AFHQv2 with FID significantly increased, shown in Table 3. Moreover, our framework enables a remarkable performance on high-resolution data sets, such as ImageNet 256 and 512, detailed see in Table 1. The qualitative analysis results are shown in Figure 3.

Transferable Residual Correction To demonstrate the transferable capability of our framework, we utilize an optimized correction function to rectify model sampling trajectories constructed by different pre-trained DMs. Concretely, we create five distinct correction functions by configuring different model channels on the architecture of EDM. For example, we configure EDM-XL with 128 channels and EDM-L with 96 channels. After optimizing those correction functions on CIFAR-10, we employ them to improve various target pre-trained DMs that also trained on CIFAR-10, results shown in Table 5.

Moreover, we further verify the transfer capability on model sampling trajectories of some classical training-free methods, for instance, DPM-Solver [46] and DEIS [78], results are displayed in Table 7. It is logical that our framework can capture the residual in images generated by training-free fast samplers, given that their sampling trajectories exhibit the same properties as the true sampling tra-

Source Pre-Trained Model: EDM [28]		Correction Function				
Target Pre-Trained Model↓	Origin FID↓	EDM-S	EDM-M	EDM-L	EDM-XL	EDM-H
NSCNv2 [65]	10.87	10.43(0.44)	10.27(0.60)	10.14(0.73)	10.01(0.86)	10.01(0.86)
DDPM [24]	3.17	3.15(0.02)	3.14(0.03)	3.09(0.08)	3.07(0.10)	3.06(0.11)
SDE VE [69]	2.55	2.52(0.03)	2.50(0.05)	2.47(0.08)	2.46(0.09)	2.46(0.09)
SDE VP [69]	2.20	2.20	2.18(0.02)	2.15(0.05)	2.15(0.05)	2.15(0.05)
EDM [28]	2.04	1.99(0.05)	1.95(0.09)	1.94(0.10)	1.93(0.11)	1.93(0.11)

Table 5. Transferable residual correction on CIFAR-10, evaluated by FID score. 'Origin FID' refers to the results obtained from the target pre-trained DMs without correction. These target pre-trained DMs include NSCNv2 [65], DDPM [24], SDE VE and VP [69]. EDM-$*$ represents the correction function constructed by the corresponding EDM backbone with different model channels (detailed seen in Table 4), referred to as the source pre-trained model. For example, EDM-XL denotes the utilization of an optimized correction function constructed by EDM-XL backbone to enhance target pre-trained DMs. In this manner, we verify that a correction function optimized on a given pre-trained DM has the ability to correct any model sampling trajectory construced by different DMs trained on the same dataset.

Correction Function	Heavy Function	
Light Function↓	EDM-XL	EDM-H
EDM-T	2.03→2.00	2.03→1.99
EDM-S	1.99→1.97	1.99→1.96
EDM-M	1.95→1.93	1.95→1.93

Table 6. Efficient correction performance evaluated by FID score. We utilize the pre-trained heavy correction functions, formulated by EDM-XL and EDM-H, to train light correction functions that formulated by EDM-T, EDM-S and EDM-M respectively. Values before → are the results obtained from baseline pre-trained DMs, and values after → are results improved by our framework.

Method	NFE 10	NFE 20	NFE 50
DPM-Solver-2 (VP)	5.28(+2)	3.02(+4)	2.69(-2)
DPM-Solver-2 (VP)*	5.12(+2)	2.94(+4)	2.65(-2)
DPM-Solver-3 (VP)	6.03(+2)	2.75(+4)	2.65(-2)
DPM-Solver-3 (VP)*	5.89(+2)	2.69(+4)	2.62(-2)
DEIS (VP)	4.17(+0)	2.86(+0)	2.57(+0)
DEIS (VP)*	4.11(+0)	2.82(+0)	2.55(+0)
DEIS (VE)	20.89(+0)	16.59(+0)	16.31(+0)
DEIS (VE)*	19.94(+0)	16.05(+0)	15.91(+0)

Table 7. FID performance on CIFAR-10 when combined with training-free fast samplers. * indicates that the results has been improved by an optimized correct function, and number in parenthesis indicates extra or less NFE used. We verify that the our framework enables to improve arbitrary model sampling trajectory formulated by training-free fast samplers.

jectory. Consequently, the experimental results further emphasize the transferable performance of our framework.

Efficient Residual Correction To avoid the slow sampling problem in DMs, we propose an efficient residual correction method. We distill correction ability from an optimized heavy correction function to a lightweight correction function, results shown in Table 6. With only a slight decline in performance, the light correction functions are 4 $\sim$ 65 $\times$times smaller than those heavy correction function.

Ablations In the ablation studies on various optimization strategies, we perform four distinct experiments to validate their effectiveness. These experiments encompass the setting of optimization metric function, ODE reverse steps, data augmentation methods and weighting functions, detailed seen in Figure 4. For the ablations on model size, we design six different correction functions with different EDM backbone channels, seen in Table 4.

6. Conclusion

In this paper, we analyze that the sampling gap resulting from score estimation and discretization errors, leading to a residual in the generated image. To remedy this, we propose a residual learning framework built upon a correction function. The optimized correction function can rectify model sampling trajectory, thereby improving the image quality. Moreover, our correction function, trained on a given pretraned DM, can enhance the arbitrary sampling trajectories of different DMs trained on the same dataset, showcasing a significant transferable capability. Empirical results clearly demonstrate the benefit of our residual learning framework.

Acknowledgement. This work was supported in part by the National Natural Science Foundation of China under grant 62172451. Chang Xu was supported in part by the Australian Research Council under Projects DP210101859 and FT230100549. This work was also supported by the National Research Foundation (NRF) grants (RS-2023-00245342) funded by the Ministry of Science and ICT (MSIT) of Korea. The AI training platform supporting this work were provided by High-Flyer AI (Hangzhou High-Flyer AI Fundamental Research Co., Ltd.). This work was also supported by the High Performance Computing Center of Central South University. This work was supported by the China Scholarship Council.

References

[1] Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982. 1, 3

[2] Fan Bao, Chongxuan Li, Jiacheng Sun, Jun Zhu, and Bo Zhang. Estimating the optimal covariance with imperfect mean in diffusion probabilistic models. *arXiv preprint arXiv:2206.07309*, 2022. 2

[3] Dmitry Baranchuk, Ivan Rubachev, Andrey Voynov, Valentin Khrulkov, and Artem Babenko. Label-efficient semantic segmentation with diffusion models. *arXiv preprint arXiv:2112.03126*, 2021. 1

[4] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018. 2

[5] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, pages 9650–9660, 2021. 2

[6] Chen-Hao Chao, Wei-Fang Sun, Bo-Wun Cheng, Yi-Chen Lo, Chia-Che Chang, Yu-Lun Liu, Yu-Lin Chang, Chia-Ping Chen, and Chun-Yi Lee. Denoising likelihood score matching for conditional score-based data generation. *arXiv preprint arXiv:2203.14206*, 2022. 2

[7] Chen-Hao Chao, Wei-Fang Sun, Bo-Wun Cheng, Yi-Chen Lo, Chia-Che Chang, Yu-Lun Liu, Yu-Lin Chang, Chia-Ping Chen, and Chun-Yi Lee. Denoising likelihood score matching for conditional score-based data generation. In *ICLR*, 2022. 2

[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 2, 6

[9] Jooyoung Choi, Jungbeom Lee, Chaehun Shin, Sungwon Kim, Hyunwoo Kim, and Sungroh Yoon. Perception prioritized training of diffusion models. In *CVPR*, pages 11472–11481, 2022. 7

[10] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. StarGAN v2: Diverse image synthesis for multiple domains. In *CVPR*, pages 8188–8197, 2020. 6

[11] Hyungjin Chung, Byeongsu Sim, Dohoon Ryu, and Jong Chul Ye. Improving diffusion models for inverse problems using manifold constraints. *NeurIPS*, 35:25683–25696, 2022. 1

[12] Giannis Daras, Mauricio Delbracio, Hossein Talebi, Alexandros G Dimakis, and Peyman Milanfar. Soft diffusion: Score matching for general corruptions. *arXiv preprint arXiv:2209.05442*, 2022. 6

[13] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. *NeurIPS*, 34:8780–8794, 2021. 1, 2, 6

[14] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016. 2

[15] Tim Dockhorn, Arash Vahdat, and Karsten Kreis. Score-based generative modeling with critically-damped langevin diffusion. In *ICLR*, 2022. 2

[16] Shanghua Gao, Pan Zhou, Ming-Ming Cheng, and Shuicheng Yan. Masked diffusion transformer is a strong image synthesizer. *arXiv preprint arXiv:2303.14389*, 2023. 1

[17] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020. 2

[18] Zheng Gu, Wenbin Li, Jing Huo, Lei Wang, and Yang Gao. Lofgan: Fusing local representations for few-shot image generation. In *CVPR*, pages 8463–8471, 2021. 2

[19] Tiankai Hang, Shuyang Gu, Chen Li, Jianmin Bao, Dong Chen, Han Hu, Xin Geng, and Baining Guo. Efficient diffusion training via min-snr weighting strategy. *arXiv preprint arXiv:2303.09556*, 2023. 2

[20] William Harvey, Saeid Naderiparizi, Vaden Masrani, Christian Weilbach, and Frank Wood. Flexible diffusion modeling of long videos. *NeurIPS*, 35:27953–27965, 2022. 1

[21] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, pages 9729–9738, 2020. 6

[22] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, pages 16000–16009, 2022. 6

[23] 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. *NeurIPS*, 30, 2017. 7

[24] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 33:6840–6851, 2020. 1, 2, 4, 6, 8

[25] Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. What makes imagenet good for transfer learning? *arXiv preprint arXiv:1608.08614*, 2016. 2

[26] Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005. 1, 2, 3

[27] Alexia Jolicoeur-Martineau, Ke Li, Rémi Piché-Taillefer, Tal Kachman, and Ioannis Mitliagkas. Gotta go fast when generating data with score-based models. *arXiv preprint arXiv:2105.14080*, 2021. 2

[28] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *NeurIPS*, 35:26565–26577, 2022. 1, 2, 3, 4, 6, 7, 8

[29] Beomsu Kim and Jong Chul Ye. Denoising MCMC for accelerating diffusion-based generative models. *arXiv preprint arXiv:2209.14593*, 2022. 1, 2

[30] Dongjun Kim, Yeongmin Kim, Wanmo Kang, and Il-Chul Moon. Refining generative process with discriminator guidance in score-based diffusion models. *arXiv preprint arXiv:2211.17091*, 2022. 1, 2, 3

[31] Dongjun Kim, Byeonghu Na, Se Jung Kwon, Dongsoo Lee, Wanmo Kang, and Il-chul Moon. Maximum likelihood training of implicit nonlinear diffusion model. *Advances in Neural Information Processing Systems*, 35:32270–32284, 2022. 6

[32] Dongjun Kim, Seungjae Shin, Kyungwoo Song, Wanmo Kang, and Il-Chul Moon. Soft truncation: A universal training technique of score-based diffusion model for high precision score estimation. In *International Conference on Machine Learning*, pages 11201–11228. PMLR, 2022. 1, 2, 3, 5, 6

[33] Dongjun Kim, Chieh-Hsin Lai, Wei-Hsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency trajectory models: Learning probability flow ode trajectory of diffusion. *arXiv preprint arXiv:2310.02279*, 2023. 2

[34] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *NeurIPS*, 34:21696–21707, 2021. 1, 2, 7

[35] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2

[36] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *CVPR*, pages 2661–2671, 2019. 2

[37] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6

[38] Nupur Kumari, Richard Zhang, Eli Shechtman, and Jun-Yan Zhu. Ensembling off-the-shelf models for gan training. In *CVPR*, pages 10651–10662, 2022. 2

[39] Shigui Li, Wei Chen, and Delu Zeng. Scire-solver: Efficient sampling of diffusion probabilistic models by score-integrand solver with recursive derivative estimation. *arXiv preprint arXiv:2308.07896*, 2023. 2, 3

[40] Daochang Liu, Qiyue Li, Anh-Dung Dinh, Tingting Jiang, Mubarak Shah, and Chang Xu. Diffusion action segmentation. In *ICCV*, pages 10139–10149, 2023. 1

[41] Gongye Liu, Haoze Sun, Jiayi Li, Fei Yin, and Yujiu Yang. Accelerating diffusion models for inverse problems through shortcut sampling. *arXiv preprint arXiv:2305.16965*, 2023. 1

[42] Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. *arXiv preprint arXiv:2202.09778*, 2022. 2

[43] Songhua Liu, Kai Wang, Xingyi Yang, Jingwen Ye, and Xinchao Wang. Dataset distillation via factorization. *Advances in neural information processing systems*, 35:1100–1113, 2022. 6

[44] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *CVPR*, pages 3730–3738, 2015. 6

[45] Cheng Lu, Kaiwen Zheng, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Maximum likelihood training for score-based diffusion odes by high order denoising score matching. In *International Conference on Machine Learning*, pages 14429–14460. PMLR, 2022. 1, 2, 3

[46] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *NeurIPS*, 35:5775–5787, 2022. 1, 2, 3, 7

[47] Calvin Luo. Understanding diffusion models: A unified perspective. *arXiv preprint arXiv:2208.11970*, 2022. 1, 2

[48] Zhengxiong Luo, Dayou Chen, Yingya Zhang, Yan Huang, Liang Wang, Yujun Shen, Deli Zhao, Jingren Zhou, and Tieniu Tan. Videofusion: Decomposed diffusion models for high-quality video generation. In *CVPR*, pages 10209–10218, 2023. 1

[49] Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *CVPR*, pages 14297–14306, 2023. 2

[50] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021. 2

[51] Alexander Quinn Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob Mcgrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. In *International Conference on Machine Learning*, pages 16784–16804. PMLR, 2022. 1

[52] Atsuhiro Noguchi and Tatsuya Harada. Image generation from small datasets via batch statistics adaptation. In *CVPR*, pages 2750–2758, 2019. 2

[53] Utkarsh Ojha, Yijun Li, Jingwan Lu, Alexei A Efros, Yong Jae Lee, Eli Shechtman, and Richard Zhang. Few-shot image generation via cross-domain correspondence. In *CVPR*, pages 10743–10752, 2021. 2

[54] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *CVPR*, pages 4195–4205, 2023. 1, 3, 6

[55] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 2

[56] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, pages 10684–10695, 2022. 1, 2, 6

[57] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *CVPR*, pages 22500–22510, 2023. 1

[58] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *NeurIPS*, 35:36479–36494, 2022. 1

[59] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022. 2

[60] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *NeurIPS*, 29, 2016. 7

[61] J Ryan Shue, Eric Ryan Chan, Ryan Po, Zachary Ankner, Jiajun Wu, and Gordon Wetzstein. 3d neural field generation using triplane diffusion. In *CVPR*, pages 20875–20886, 2023. 1

[62] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015. 1

[63] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2022. 2

[64] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *NeurIPS*, 32, 2019. 1, 2, 3

[65] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *NeurIPS*, 33:12438–12448, 2020. 1, 3, 6, 8

[66] 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*, pages 574–584. PMLR, 2020. 1, 3

[67] Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. *NeurIPS*, 34:1415–1428, 2021. 1, 2, 3, 5

[68] Yang Song, Liyue Shen, Lei Xing, and Stefano Ermon. Solving inverse problems in medical imaging with score-based generative models. *arXiv preprint arXiv:2111.08005*, 2021. 1

[69] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*, 2021. 1, 2, 3, 4, 6, 8

[70] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023. 2, 3, 6

[71] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *CVPR*, pages 7167–7176, 2017. 2

[72] Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space. *NeurIPS*, 34:11287–11302, 2021. 1, 2

[73] Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International conference on machine learning*, pages 1747–1756. PMLR, 2016. 6

[74] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011. 3

[75] Zike Wu, Pan Zhou, Kenji Kawaguchi, and Hanwang Zhang. Fast diffusion model, 2023. 6

[76] Jamie Wynn and Daniyar Turmukhambetov. Diffusionerf: Regularizing neural radiance fields with denoising diffusion models. In *CVPR*, pages 4180–4189, 2023. 1

[77] Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *CVPR*, pages 3712–3722, 2018. 2

[78] Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. *ICLR*, 2023. 1, 2, 3, 7

[79] Qinsheng Zhang, Jiaming Song, and Yongxin Chen. Improved order analysis and design of exponential integrator for diffusion models sampling. *arXiv preprint arXiv:2308.02157*, 2023. 2

[80] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, pages 586–595, 2018. 6

[81] Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. Unipc: A unified predictor-corrector framework for fast sampling of diffusion models. *arXiv preprint arXiv:2302.04867*, 2023. 2