This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Fixed Point Diffusion Models

Xingjian Bai *
University of Oxford
xingjianbai@gmail.com

Luke Melas-Kyriazi *†
University of Oxford
lukemk@robots.ox.ac.uk

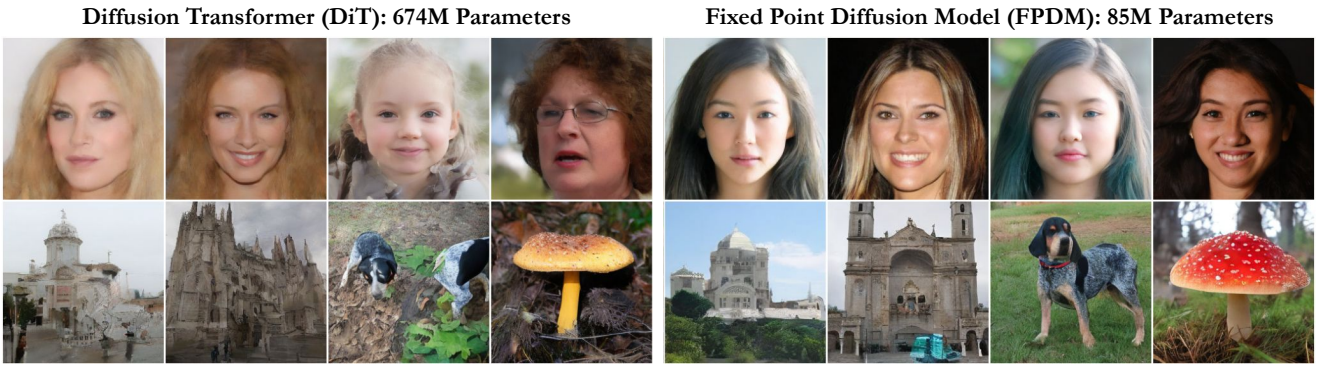


Figure 1. **Fixed Point Diffusion Model (FPDM)** is a novel and highly efficient approach to image generation with diffusion models. FPDM integrates an implicit fixed point layer into a denoising diffusion model, converting the sampling process into a sequence of fixed point equations. Our model significantly decreases model size and memory usage while improving performance in settings with limited sampling time or computation. We compare our model, trained at a 256 $\times$ 256 resolution against the state-of-the-art DiT [38] on four datasets (FFHQ, CelebA-HQ, LSUN-Church, ImageNet) using compute equivalent to 20 DiT sampling steps. FPDM (right) demonstrates enhanced image quality with 87% fewer parameters and 60% less memory during training.

Abstract

We introduce the Fixed Point Diffusion Model (FPDM), a novel approach to image generation that integrates the concept of fixed point solving into the framework of diffusion-based generative modeling. Our approach embeds an implicit fixed point solving layer into the denoising network of a diffusion model, transforming the diffusion process into a sequence of closely-related fixed point problems. Combined with a new stochastic training method, this approach significantly reduces model size, reduces memory usage, and accelerates training. Moreover, it enables the development of two new techniques to improve sampling efficiency: reallocating computation across timesteps and reusing fixed point solutions between timesteps. We conduct extensive experiments with state-of-the-art models on ImageNet, FFHQ, CelebA-HQ, and LSUN-Church, demonstrating substantial improvements in performance and efficiency. Compared to the state-of-the-art DiT model [38], FPDM contains 87% fewer parameters, consumes 60% less memory during training, and improves image generation quality in situations where sampling computation or time is limited. Our code and pretrained models are available at https://lukemelas.github.io/fixed-point-diffusion-models/.

1. Introduction

The field of image generation has experienced significant recent advancements driven by the development of large-scale diffusion models [23, 37, 38, 41, 47, 48]. Key to these advancements have been increased model size, computational power, and the collection of extensive datasets [4, 12, 16, 25, 45, 46, 54], collectively contributing to a marked improvement in generation performance.

Despite these strides, the core principles of diffusion networks have remained largely unchanged since their development [23]. They typically consist of a fixed series of neural network layers, either with a UNet architecture [42] or, more recently, a vision transformer architecture [14, 51]. However, as diffusion models are increasingly deployed in production, especially on mobile and edge devices, their large size and computational costs pose significant challenges.

This paper introduces the Fixed Point Diffusion Model (FPDM), which integrates an implicit fixed point solving layer into the denoising network of a diffusion model. In

*Equal Contribution.
†Corresponding author.

contrast to traditional networks with a fixed number of layers, FPDM is able to utilize a variable amount of computation at each timestep, with the amount of computation directly influencing the accuracy of the resulting solutions. This fixed point network is then applied sequentially, as in standard diffusion models, to progressively denoise a data sample from pure Gaussian noise.

FPDM offers efficiency gains at two levels of granularity: that of individual timesteps and that of the entire diffusion process. First, at the timestep level, it provides:

1. A substantial reduction in parameter count compared to previous networks (87% compared to DiT [38]).
2. Reduced memory usage during both training and sampling (60% compared to DiT [38]).

Second, at the diffusion process level, it provides:

1. The ability to smoothly distribute or reallocate computation among timesteps. This contrasts with all previous diffusion models, which must perform a full forward pass at every sampling timestep.
2. The capacity to reuse solutions from one fixed-point layer as an initialization for the layer in the subsequent timestep, further improving efficiency.

Our fixed-point network thereby delivers immediate benefits, in the form of reduced size and memory (Sec. 3.2), and further benefits when integrated into the diffusion process, in the form of increased flexibility during sampling (Sec. 3.3).

To realize these benefits, it is imperative to train our models using an efficient differentiable fixed-point solver. Although several implicit training methods exist in the literature [5, 15, 18], we find them to be unstable or underperformant in our setting. Hence, we develop a new training procedure named Stochastic Jacobian-Free Backpropagation (S-JFB) (Sec. 3.4), inspired by Jacobian-Free Backpropagation (JFB) [15]. This procedure is stable, highly memory-efficient, and surpasses standard JFB in performance.

We demonstrate the efficacy of our method through extensive experiments (Sec. 4) on four popular image generation datasets: LSUN-Church [54], CelebA-HQ [25], FFHQ [4], and ImageNet [12]. FPDM excels over standard diffusion models when computational resources during sampling are limited. Finally, detailed analysis and ablation studies (Sec. 5) demonstrate the efficacy of our proposed network, sampling techniques, and training methods.

2. Related Work

Diffusion Models (DMs). Diffusion models [2, 23], or score-based generative models [48, 49], are the source of tremendous recent progress in image generation. They learn to reverse a Markovian noising process using a denoiser parametrized by a neural network, traditionally a U-Net [42]. The denoising paradigm can be seen as the discretization of a stochastic differential equation in a continuous domain [50]. Later work equipped DMs with different sampling methods [34, 47, 53] and applied conditional control from multiple modalities [13, 36, 49]. Recently, DMs with transformer-based architectures (DiTs) were shown to be highly effective [38]; FPDM builds upon the DiT architecture.

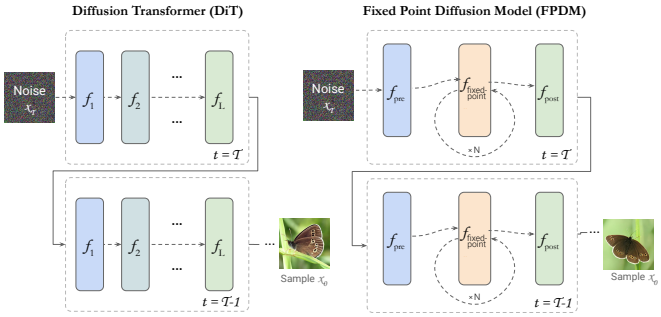


Figure 2. **The architecture of FPDM compared with DiT.** FPDM keeps the first and last transformer block as pre and post processing layers and replaces the explicit layers in-between with an implicit fixed point layer. The diffusion sampling process involves solving many of these fixed point layers in sequence, which enables the development of new techniques such as timestep smoothing (Sec. 3.3) and solution reuse (Sec. 3.3).

The heavy memory and computation requirements of DMs scale up quadratically with the image resolution and linearly with the number of sampling timesteps. To reduce training cost, LDM [41] proposes to downsample images with a pre-trained Variational Autoencoder [29] and perform denoising in latent space. However, the inference cost of DMs is still considered their primary drawback.

Implicit Networks and Deep Equilibrium Models. Whereas traditional neural networks calculate outputs by performing a pass through a stack of layers, implicit neural networks define their outputs by the solutions of dynamic systems. Specifically, Deep Equilibrium Models (DEQs) [5, 17] define their output by the fixed point of an equilibrium layer f_θ. The equilibrium state of DEQs, z^*, is equivalent to the output of an infinite-depth, weight-sharing explicit neural network: $\lim_{k\to\infty} f_\theta(z^k) = f_\theta(z^*) = z^*$. In its forward pass, the equilibrium state z^* can be computed by applying solvers like Broyden's method [9] or Anderson's acceleration [3]. In the backward pass, one can implicitly differentiate through the equilibrium state z^*, or use one of the recently-proposed accelerated training methods [15, 18]. Applications of DEQs include optical flow [7], image segmentation [6] and inverse imaging problems [20].

Recent Work Combining Diffusion and DEQs. In the past year, two works have merged DMs and DEQs. Differently from our proposal, these approaches tried to convert the *entire diffusion process* into a *single* fixed point equation. [39] considers the entire diffusion trajectory as a single sample and solves for the fixed point of the trajectory, converting the sequential diffusion process into a parallel one.

[19] distills a pretrained diffusion model into a single-step DEQ. These works are exciting but come with their own drawbacks: the former is an inference-time technique that consumes significantly more memory than standard ancestral sampling, while the latter requires a pretrained diffusion model and has not scaled to datasets larger than CIFAR-10.

3. Methods

3.1. Preliminaries

Implicit Neural Networks. The neural network layer is the foundational building block of deep learning. While early neural networks used only a few layers [30, 32], modern networks such as large-scale transformers [14, 51] often consist of dozens of layers connected by residual blocks. Typically, these layers share a similar internal structure and dimensionality, with each having distinct set of parameters. In essence, most networks are defined *explicitly*: their operations are precisely defined by their layer weights. Running a forward pass always entails processing inputs with the entire set of layers.

On the other hand, *implicit* models define the function or procedure to be computed by the network, rather than the exact sequence of operations. This category includes models that integrate differential equation solvers (Neural ODE/CDE/SDEs; [10, 27, 28]), as well as models incorporating fixed point solvers (fixed point networks or DEQs; [5]). Our proposed FPDM belongs to this latter group.

Differentiable Fixed Point Solving. Given a function f on X, a fixed point of f is $x^* \in X$ such that $f(x^*) = x^*$. The computation of fixed points has been the subject of centuries of mathematical study [52], with the existence and uniqueness of a system's fixed points often proved with the Banach fixed-point theorem and its variants [1, 21, 24].

In our case, $f = f_\theta$ is a differentiable function parametrized by θ, and we are interested in both solving for the fixed point and backpropagating through it. The simplest solving method is *fixed point iteration*, which iteratively applies f_θ until convergence to x^*. Under suitable assumptions, iteration converges linearly to the unique fixed point of an equilibrium system (Thrm 2.1 in [15]). Alternative methods found throughout the literature include Newton's method, quasi-Newton methods such as Broyden's method, and Anderson's acceleration. In these cases, one can analytically backpropagate through x^* via implicit differentiation [15]. However, these methods can come with significant memory and computational costs. Recently, a new iterative solving method denoted Jacobian-Free Backpropagation (JFB) was introduced to circumvent the need for complex and costly implicit differentiation; we discuss and extend upon this method in Sec. 3.4.

3.2. Fixed Point Denoising Networks

Our proposed fixed-point denoising network (Fig. 2) integrates an implicit fixed-point layer into a diffusion transformer. The network consists of three stages: 1) explicit timestep-independent preprocessing layers $f_{\text{pre}}^{(1:l)} : X \to X$, 2) a implicit timestep-conditioned fixed-point layer $f_{\text{fp}} : X \times X \times T \to X$, and 3) explicit timestep-independent postprocessing layers $f_{\text{post}}^{(1:l)} : X \to X$. The function f_{fp} takes as input both the current fixed-point solution x and a value $\tilde{x}$ called the *input injection*, which is the projected output of the preceding explicit layers. One can think of f_{fp} as a map $f_{\text{fp}}^{(\tilde{x},t)} : X \to X$ conditional on the input injection and timestep, for which we aim to find a fixed point. The network processes an input $x_{\text{input}}^{(t)}$ as follows:

$$x_{\text{pre}}^{(t)} = f_{\text{pre}}^{(1:l)}(x_{\text{input}}^{(t)}) \tag{1}$$

$$\tilde{x}^{(t)} = \text{projection}(x_{\text{pre}}^{(t)}) \qquad \textit{input injection} \tag{2}$$

$$x^{*(t)} = f_{\text{fp}}(x^{*(t)}, \tilde{x}^{(t)}, t) \qquad \textit{via fixed point solving} \tag{3}$$

$$x_{\text{post}}^{(t)} = f_{\text{post}}^{(1:l)}(x^{*(t)}) \tag{4}$$

The output $x_{\text{post}}^{(t)}$ is used to compute the loss (during training) or the input $x_{\text{input}}^{(t-1)}$ to the next timestep (during sampling).

Whereas explicit networks consume a fixed amount of computation, this implicit network can adapt based on the desired level of accuracy or even on the difficulty of the input. In this way, it unlocks a new tradeoff between computation and accuracy. Moreover, since the implicit layer replaces a large number of explicit layers, it significantly decreases its size and memory consumption.

Finally, note that our denoising network operates in latent space rather than pixel space. That is, we apply a Variational Autoencoder [29, 41] to encode the input image into latent space and perform all processing in latent space.

3.3. Fixed Point Diffusion Models (FPDM)

FPDM incorporates the fixed point denoising network proposed above into a denoising diffusion process.

We assume the reader is already familiar with the basics of diffusion models and provide only a brief summary; if not, we provide an overview in the Supplementary Material. Diffusion models learn to reverse a Markovian noising process in which a sample $X_0 \sim q(X_0)$ from a target data distribution $q(X_0)$ is noised over a series of timesteps $t \in [0, T]$. The size of each step of this process is governed by a variance schedule $\{\beta_t\}_{t=0}^T$ as $q(X_t|X_{t-1}) = \mathcal{N}(X_t; \sqrt{1-\beta_t}X_{t-1}, \beta_t\mathbf{I})$. where each $q(X_t|X_{t-1})$ is a Gaussian distribution. We learn the distribution $q(X_{t-1}|X_t)$ using a network $s_\theta(X_{t-1}|X_t) \approx q(X_{t-1}|X_t)$, which in our case is a fixed point denoising network. The generative process then begins with a sample from the noise distribu-

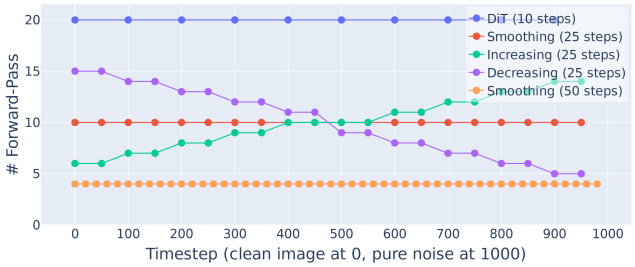


Figure 3. **Illustration of Transformer Block Forward Pass Allocation in FPDM and DiT.** Since DiT has to perform full forward passes at each timestep, under limited compute, it can only denoise at a few timesteps with large gaps. FPDM achieves a more balanced distribution through smoothing, thereby reducing the discretization error. Additionally, FPDM offers the flexibility to adjust forward pass allocation per timestep with heuristics like *Increasing* and *Decreasing*. Refer to Section 3.3 for details.

tion $q(X_T)$ and denoises it over a series of steps to obtain a sample from the target distribution $q(X_0)$.

The primary drawback of diffusion models as a class of generative models is that they are relatively slow to sample. As a result, during sampling, it is very common to only use a small subset of all diffusion timesteps and take correspondingly larger sampling steps; for example, one might train with 1000 timesteps and then sample images with as few as $N = 5, 10$, or 20 timesteps.

Naturally, one could replace the explicit denoising network inside a standard diffusion model with a fixed point denoising network, and make no other changes; this would immediately reduce model size and memory usage, as discussed previously. However, we can *further* improve efficiency during sampling by exploiting the fact that we are solving a *sequence* of related fixed point problems across all timesteps, instead of a single one. We present two opportunities for improvement: smoothing/reallocating computation across timesteps and reusing solutions.

Smoothing Computation Across Timesteps. The flexibility afforded by fixed-point solving enables us to allocate computation between timesteps in a way that is not possible with traditional diffusion models. For a given computational budget for sampling, we can reduce the the number of forward passes (i.e. number of fixed point iterations) used per timestep in order to use more timesteps over the sampling process (see Fig. 3). In other words, our implicit model can effectively "smooth out" the computation over more timesteps. By contrast, with explicit models such as DiT, the amount of computation directly determines the number of timesteps, since a full forward pass is needed at each timestep. Indeed, we find that when the amount of compute is relatively limited, it is highly beneficial to smooth out the compute among more timesteps than would be done with a traditional model. The effectiveness of smoothing is shown empirically in section Sec. 5.1.

Reallocating Computation Across Timesteps Beyond smoothing out computation over timesteps, FPDM enables one to vary the number of forward passes at each timestep, thereby dynamically controlling the solving accuracy at different stages of the denoising process. This capability enables the implementation of diverse heuristics. For example, we can allocate a greater number of iterations at the start ("decreasing") or the end ("increasing") of the diffusion process (see Fig. 3). Additionally, FPDM supports adaptive allocations of forward passes; further discussions can be found in the supplementary material (D,E).

Reusing Solutions. The convergence speed of fixed point solving meaningfully depends on the initial solution provided as input. A poor guess of the initial solution would make the convergence slower. Considering each timestep independently, the usual approach would be to initialize the fixed-point iteration of each timestep using the output of the (explicit) layers. However, considering sequential fixed point problems in the diffusion process, we can reuse the solution from the fixed point layer at the previous timestep as the initial solution for the next timestep. Formally, we can initialize the iteration in Eq. (3) with $x^{*(t-1)}$ rather than with $x^{(t)}_{pre}$.

The intuition for this idea is that adjacent timesteps of the diffusion process only differ by a small amount of noise, so their fixed point problems should be similar. Hence, the solutions of these problems should be close, and the solution of the previous timestep would be a good initialization for the current timestep. A similar idea was explored in [8], which used fixed point networks for optical flow estimation.

3.4. Stochastic Jacobian-Free Backpropagation

The final unresolved aspect of our method is how to train the network, i.e. how to effectively backpropagate through the fixed point solving layer. While early work on DEQs used expensive (Jacobian-based) implicit differentiation techniques [5], recent work has found more success using approximate and inexpensive (Jacobian-free) gradients [15].

Concretely, this consists of first computing an approximate fixed point (either via iteration or Broyden's method) *without storing any intermediate results for backpropagation*, and then taking a single additional fixed point step while storing intermediate results for backpropagation in the standard way. During the backward pass, the gradient is only computed for the final step, and so it is referred to as a "1-step gradient" or Jacobian-Free Backpropagation (JFB). [1]

Formally, this approximates the gradient of the loss $\mathcal{L}$ with respect to the parameters θ of the implicit layer f_{fp} with

[1]We note that this process has been referred to in the literature by many names, including the 1-step gradient, phantom gradient, inexact gradient, and Jacobian-Free backpropagation.

fixed point $x^{*(t)}$ by

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial x^{*(t)}} \left(I - \frac{\partial f_{\text{fp}}(x^{*(t)}, \tilde{x}^{(t)}, t)}{\partial x^{*(t)}} \right) \frac{\partial f_{\text{fp}}(x^{*(t)}, \tilde{x}^{(t)}, t)}{\partial \theta} \approx \frac{\partial \mathcal{L}}{\partial x^{*(t)}} \frac{\partial f_{\text{fp}}(x^{*(t)}, \tilde{x}^{(t)}, t)}{\partial \theta}$$

The equality above is a consequence of the Implicit Function Theorem [43] and the approximation simply drops the inverse Jacobian term. This simplification is rigorously justified by Theorem 3.1 in [15] under appropriate assumptions.

Despite the simplicity of the 1-step gradient, we found that it performed poorly in large-scale experiments. To improve performance without sacrificing efficiency, we use a new stochastic approach to compute approximate gradients.

Our approach (Algorithm 1) samples random variables $n \sim U[0, N]$ and $m \sim U[1, M]$ at each training step. During the forward pass, we perform n fixed point iterations without gradient to obtain an approximate solution,[2], and then we perform m additional iterations with gradient. During the backward pass, we backpropagate by unrolling only through the last m iterations. The constants N and M are hyperparameters that refer to the maximum number of training iterations without and with gradient, respectively. When sampling, the number of iterations used at each timestep is flexible and can be chosen independently of N or M. Compared to the 1-step gradient, our method consumes more memory and compute because it backpropagates through multiple unrolled iterations rather than a single iteration. However, it is still drastically more efficient than either implicit differentiation or using traditional explicit networks, and it significantly outperforms the 1-step gradient in our experiments (Tab. 5).

Algorithm 1 Stochastic Jacobian-Free Backpropagation

Input hidden states x, timestep t

function FORWARD(x)
  $\tilde{x} \leftarrow$ PROJ(x) # input injection
  for n iterations drawn uniformly from 0 to N **do**
    $x \leftarrow$ FORWARDPASSWITHOUTGRAD($x, \tilde{x}, t$)
  for m iterations drawn uniformly from 1 to M **do**
    $x \leftarrow$ FORWARDPASSWITHGRAD($x, \tilde{x}, t$)
  BACKPROP($loss, x$)
  return x

4. Experiments

4.1. Experimental Setup

Model The architecture of FPDM is based on the current state-of-the-art in generative image modeling, DiT-XL/2 [38], which serves as a strong baseline in our experiments. Adhering to the architecture in [38], we operate in latent space using the Variational Autoencoder from [29, 41]. In addition, we have equipped both the baseline DiT and our FPDM with two advances from the recent diffusion literature: (1) training to predict velocity rather than noise [44], and (2) modifying the denoising schedule to have zero terminal signal-to-noise ratio [33]. We include these changes to show that our improvements are orthogonal to other improvements made in the diffusion literature.

Our network consists of three sets of layers: pre-layers, an implicit fixed point layer, and post-layers. All layers have the same structure and 24M parameters, except the implicit layer has an additional projection for input injection. Through empirical analysis, we find that a single pre/post layer can achieve strong results (see Sec. 5.3). Consequently, the number of parameters in our full network is only 86M, markedly lower than 674M parameters in the standard DiT XL/2 model, which has 28 explicit layers.

Training We perform experiments on four diverse and popular datasets: Imagenet, CelebA-HQ, LSUN Church, and FFHQ. All experiments are performed at resolution 256. The ImageNet experiments are class-conditional, whereas those on other datasets are unconditional. For a fair comparison, we train our models and baseline DiT models for the same amount of time using the same computational resources. All models are trained on 8 NVIDIA V100 GPUs; the models for the primary experiments on ImageNet are trained for four days (equivalent to 400,000 DiT training steps), while those for the other datasets and for the ablation experiments are trained for one day (equivalent to 100,000 DiT steps). We train using Stochastic JFB with $M = N = 12$ and provide an analysis of this setup in Sec. 5.4.

The ImageNet experiments are class-conditional, whereas those on other datasets are unconditional. For ImageNet, following DiT, we train using class dropout of 0.1, but we compute quantitative results without classifier-free guidance. We train with a total batch size of 512 and learning rate 1e$-$4. We use a linear diffusion noise schedule with $\beta_{\text{start}} = 0.0001$ and $\beta_{\text{end}} = 0.02$, modified to have zero terminal SNR [33]. We use v-prediction as also recommended by [33]. Following DiT [38], we learn the variance σ along with the velocity v.

Finally, with regard to the evaluations, all evaluations were performed using 50000 images (FID-50K) except those in Tab. 7 and Fig. 6, which were computed using 1000 images due to computational constraints.

4.2. Sampling Quality and Cost Evaluation

To measure image quality, we employ the widely-used Frechet Inception Distance (FID) 50K metric [22]. To measure the computational cost of sampling, previous studies

[2]By "without gradient", we mean that these iterations do not store any intermediate results for backpropagation, and they are not included in gradient computation during the backward pass.

Blocks	*Model*	*FID (DDPM)*	*FID (DDIM)*	*Params.*	*Training Memory*
140	DiT	148.0	110.0	674M	25.2 GB
	FPDM	**85.8**	**33.9**	**85M**	**10.2 GB**
280	DiT	80.9	35.2	674M	25.2 GB
	FPDM	**43.3**	**22.4**	**85M**	**10.2 GB**
560	DiT	37.9	**16.5**	674M	25.2 GB
	FPDM	**26.1**	19.6	**85M**	**10.2 GB**

Table 1. **Quantitative Results on ImageNet.** Despite having 87% fewer parameters and using 60% less memory during training, FPDM outperforms DiT [38] at 140 and 280 transformer block forward passes and achieves comparable performance at 560 passes.

on diffusion model sampling have counted the number of function evaluations (NFE) [26, 35, 55]. However, given the implicit nature of our model, a more granular approach is required. In our experiments, both implicit and explicit networks consist of transformer blocks with identical size and structure, so the computational cost of each sampling step is directly proportional to the number of transformer block forward passes executed; the total cost of sampling is the product of this amount and the total number of timesteps.[3] As a result, we quantify the sampling cost in terms of *total transformer block forward passes*. [4]

Dataset	*Model*	*FID*	*Dataset*	*Model*	*FID*
CelebA-HQ	DiT	65.2	FFHQ	DiT	58.1
	FPDM	**11.1**		FPDM	**18.2**
LSUN-Church	DiT	65.6	ImageNet	DiT	80.9
	FPDM	**22.7**		FPDM	**43.3**

Table 2. **Quantitative Results Across Four Datasets.** FPDM consistently outperforms DiT [38] on CelebA-HQ, FFHQ, LSUN-Church, and Imagenet with 280 transformer block forward passes. All models are trained and evaluated at resolution 256px using the same amount of compute and identical hyperparameters.

4.3. Results

In Tab. 1, we first present a quantitative comparison of our proposed FPDM against the baseline DiT, under different amounts of sampling compute. Notably, given 140 and 280 transformer block forward passes, our best model significantly outperforms DiT, with the widest performance gap

[3]To be predise the implicit layer includes an extra projection for input injection, but this difference is negligible.

[4]For example, sampling from a FPDM with one pre/post-layer and 26 fixed point iterations across S timesteps requires the same amount of compute/time as a FPDM with two pre/post layers and 10 iterations using $2S$ timesteps; this computation cost is also the same as that of a traditional DiT with 28 layers across S timesteps. Formally, the sampling cost of FPDM is calculated by $(n_{\text{pre}} + n_{\text{iters}} + n_{\text{post}}) \cdot S$ where n_{pre} and n_{post} are the number of pre- and post-layers, n_{iters} the number of fixed point iterations, and S the number of sampling steps.

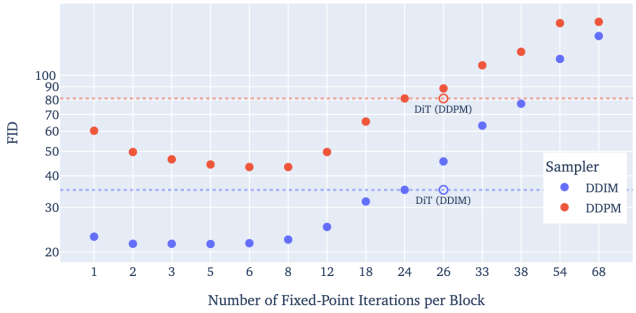


Figure 4. **Timestep smoothing significantly improves performance.** Given the same amount of sampling compute (280 transformer blocks), FPDM enables us to allocate computation among more or fewer diffusion timesteps, creating a tradeoff between the number of fixed-point solving iterations per timestep and the number of timesteps in the diffusion process (See Sec. 3.3). Here we explore the performance of our model on ImageNet with fixed point iterations ranging from 1 iteration (across 93 timesteps) to 68 iterations (across 4 timesteps). Each timestep also has 1 pre- and post-layer, so sampling with k iterations utilizes $k + 2$ blocks of compute per timestep. The circle and dashed lines show the performance of the baseline DiT-XL/2 model with 28 layers, which in our formulation corresponds to smoothing over 26 iterations. Although our model is slightly worse than DiT at 26 iterations, it significantly outperforms DiT when smoothed across more timesteps, demonstrating the effectiveness of timestep smoothing.

Model	Num. Params	FID
DiT-XL/2 (3 layers)	86M	350.6
DiT-B/2 (11 layers)	130M	55.9
FPDM-XL/2	86M	**31.5**

Table 3. **Comparison of models with similar parameter counts.** All models use approximately the same amount of inference-time compute, equivalent to 280 blocks of a DiT-XL/2-sized transformer.

given the most limited compute. Our method's improvements are orthogonal to those gained from using better samplers; our model effectively lowers the FID score with both DDIM and DDPM. At 560 forward passes, our method outperforms DiT with DDPM but not DDIM, and for more than 560 it is outperformed by DiT. Note that the number of parameters in FPDM is only 12.9% of that in DiT, and it consumes 60% less memory during training (reducing memory from 25.2 GB to only 10.2 GB at a batch size of 64).

Tab. 2 extends the comparison between FPDM and DiT to three additional image datasets: FFHQ, CelebA-HQ, and LSUN-Church. Our findings are consistent across these datasets, with FPDM markedly improving the FID score despite being nearly one-tenth the size of DiT.

Table 3 compares the results of our method to DiT models with similar parameter counts, rather than much larger models. We significantly outperform these models.

Fig. 1 shows qualitative results of our model compared to DiT. All images are computed using the same random seeds and classes using a classifier-free guidance scale 4.0 and 560 transformer block forward passes (20 timesteps for DiT). FPDM uses 8 fixed point iterations per block with timestep smoothing. Our model produces sharper images with more

Figure 5. **Qualitative Results for Smoothing Computation Across Timesteps**. We show visual results of FPDM using different numbers of fixed point solving iterations, while keeping the total amount of sampling compute fixed (560 transformer blocks). Our method demonstrates similar performance compared to the baseline with 20 to 30 iterations per timestep and superior generation quality with 4 to 8 iterations, as reflected quantitatively in Fig. 4.

Train Iters. (M, N)	*3*	*6*	*12*	*24*
FID	**43.0**	43.2	61.5	567.6

Table 4. **Performance For Varying Numbers of Fixed Point Iterations in Training.** This table compares various choices of M and N values in Algorithm 1. They represent a tradeoff between training speed and fixed point convergence accuracy. Results indicate that the optimal values for M and N range from 3 to 6.

Train iters without grad (N)	*Method*	*FID*
6	JFB (1-Step Grad)	567.6
	Multi-Step JFB	48.2
	Stochastic JFB	**43.2**
12	JFB (1-Step Grad)	567.6
	Multi-Step JFB	79.9
	Stochastic JFB	**61.5**

Table 5. **Performance of Stochastic Jacobian-Free Backpropagation (S-JFB) compared to JFB (1-step gradient).** We find that 1-step gradient, the most common method for training DEQ [5] models, struggles to optimize models on the large-scale ImageNet dataset, whereas a multi-step version of it performs well and our stochastic multi-step version performs (S-JFB) even better. The 1-step gradient always unrolls with gradient through a single iteration ($M = 1$) of fixed point solving, whereas the stochastic version unrolls though $m \sim U(1, M)$ iterations for $M = 12$.

detail, likely due to its ability to spread the computation among timesteps, as discussed in Sec. 5.1.

5. Analysis and Ablation Studies

5.1. Smoothing Computation Across Timesteps

In Fig. 4 and 5, we examine the effect of smoothing described in Sec. 3.3. We sample across a range of iterations and timesteps, keeping the total cost (i.e. the number of transformer block forward passes) constant. This explores the trade-off between the fixed point solving at each timestep and the discretization of the entire diffusion process.

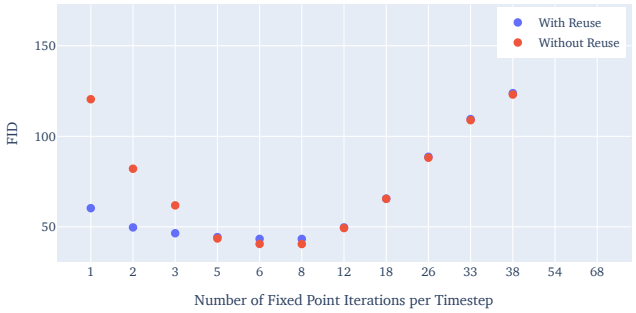
Results show that balancing iterations and timesteps is crucial to achieving high performance. Intuitively, when using very few iterations per timestep, the model fails to converge adequately at each step, and the resulting error compounds. Conversely, allocating many iterations to a few timesteps results in unnecessary computation on already converged solving iterations, resulting in large discretization errors arising from larger gaps between timesteps. An ideal strategy uses just enough fixed-point iterations to achieve satisfactory solutions, thereby maximizing the number of timesteps. For instance, with 280 transformer block forward passes, using 4 and 8 iterations per timestep is optimal.

5.2. Reusing Solutions

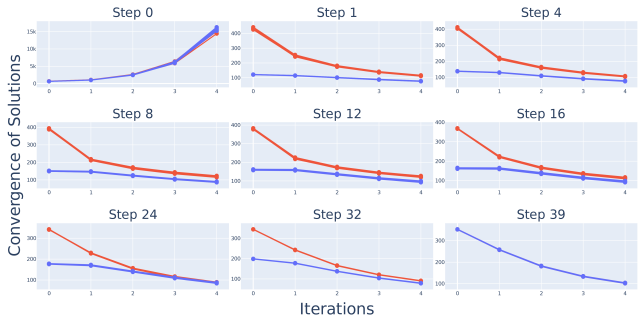
As described in Sec. 3.3, we explore reusing the fixed point solution from each timestep to initialize the subsequent step. In Fig. 6a, we see that reusing makes a big difference when performing a small number of iterations per timestep and a negligible difference when performing many iterations per timestep. Intuitively, reusing solutions reduces the number of iterations needed at each timestep, so it improves the performance when the number of iterations is severely limited. Fig. 6b and Fig. 6c illustrate the functionality of reusing by examining at the individual timestep level. For each timestep t, we use the difference between the last two fixed point iterations, δ_t, as an indicator for convergence. Reusing decreases δ_t for all timesteps except a few noisiest steps, and reusing is most effective at less noisy timesteps. This observation aligns with our intuition: Adjacent timesteps with less noise tend to have highly similar corresponding fixed point systems, where reusing is more effective.

5.3. Pre/Post Layers

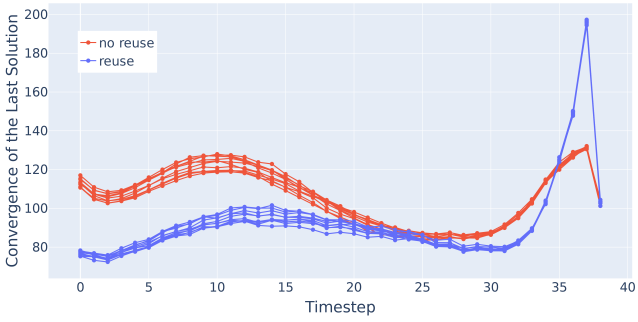
One of the many ways FPDM differs from prior work on DEQs is that we include explicit layers before and after the implicit fixed point layer. In Fig. 7, we conduct an ablation



(a) Performance improvement from reusing solutions across timesteps.



(b) F.p. convergence for nine timesteps, with and without reuse.



(c) Convergence at iteration across timesteps, with and without reuse.

Figure 6. **The Impact of Reuse on Fixed Point Accuracy.** In (a), we examine sampling performance with and without reusing solutions for different numbers of iterations per timestep; reusing considerably helps when using a few iterations per timestep. In (b) and (c), we examine the convergence of individual timesteps. Reuse delivers particularly large benefits for smaller (less-noisy) timesteps. Note that these plots contain 10 lines as they are plotted for 10 random batches of 32 images from ImageNet.

analysis, training networks with 0, 1, 2, and 4 pre/post layers. We see that using at least 1 explicit pre/post layer is always better than 0. For small compute budgets it is optimal to use 1 pre/post layer, and for larger budgets it is optimal to use 2 or 4. Broadly, we observe that using more explicit layers limits flexibility thereby reducing performance at lower compute budgets, but improves performance at higher budgets.

5.4. Training Method

In Tab. 4, we compare versions of Stochastic Jacobian-Free Backpropagation with different values of M and N, the upper bounds on the number of training iterations with and without gradient. M and N reflect a training-time tradeoff

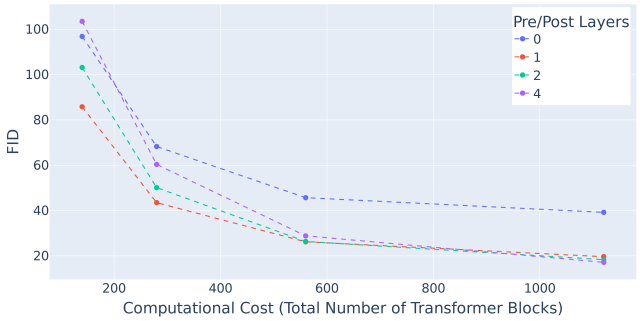


Figure 7. **Performance of Different Number of Pre/Post Layers.** We find that using at least one pre/post layer is always better than none; fewer explicit layers perform better on small compute budgets, whereas more explicit layers can better leverage large budgets.

between speed and fixed point convergence. As M and N increase, each training step contains more transformer block forward and backward passes on average; the fixed point approximations become more accurate, but each step consumes more time and memory. We find the optimal values of M and N are quite low: 3 or 6. Using too many iterations (e.g. 24) is detrimental as it slow down training.

In Tab. 5, we compare to JFB (also called 1-step gradient), which has been used in prior work on DEQs [15], and a multi-step variant of it. We find that training with multiple steps is essential to obtaining good results, and that using a stochastic number of steps delivers further improvements.

5.5. Limitations

The primary limitation of our model is that it performs worse than the fully-explicit DiT model when sampling computation and time are not constrained. The performance gains from our model in resource-constrained settings stem largely from smoothing and reusing, but in scenarios with saturated timesteps and iterations, the efficacy of these techniques is reduced. In such cases, our network resembles a transformer with weight sharing [31, 40], which underperform vanilla transformers. Hence, we do not expect to match the performance of DiT, which has 8× more parameters, when sampling with an unlimited amount of resources.

6. Conclusions

We introduce FPDM, a pioneering diffusion model characterized by fixed point implicit layers. Compared to traditional Diffusion Transformers (DiT), FPDM significantly reduces model size and memory usage. In the context of diffusion sampling, FPDM enables us to develop new techniques such as solution reusing and timestep smoothing, which give FPDM enhanced flexibility in computational allocation during inference. This flexibility makes it particularly effective in scenarios where computational resources are constrained. Future work could explore new ways of leveraging this flexibility as well as scaling to larger datasets such as LAION-5B [46].

References

[1] Ravi P Agarwal, Maria Meehan, and Donal O'regan. *Fixed point theory and applications*. Cambridge university press, 2001. 3

[2] Namrata Anand and Tudor Achim. Protein structure and sequence generation with equivariant denoising diffusion probabilistic models. *arXiv preprint arXiv:2205.15019*, 2022. 2

[3] Donald G. Anderson. Iterative procedures for nonlinear integral equations. *J. ACM*, 12(4):547–560, 1965. 2

[4] Haoran Bai, Di Kang, Haoxian Zhang, Jin-shan Pan, and Linchao Bao. Ffhq-uv: Normalized facial uv-texture dataset for 3d face reconstruction. *ArXiv*, abs/2211.13874, 2022. 1, 2

[5] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 688–699, 2019. 2, 3, 4, 7

[6] Shaojie Bai, Vladlen Koltun, and J Zico Kolter. Multiscale deep equilibrium models. *Advances in Neural Information Processing Systems*, 33:5238–5250, 2020. 2

[7] Shaojie Bai, Zhengyang Geng, Yash Savani, and J Zico Kolter. Deep equilibrium optical flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 620–630, 2022. 2

[8] Shaojie Bai, Zhengyang Geng, Yash Savani, and J. Zico Kolter. Deep equilibrium optical flow estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 610–620. IEEE, 2022. 4

[9] C. G. Broyden. A class of methods for solving nonlinear simultaneous equations. *Mathematics of Computation*, 19 (92):577–593, 1965. 2

[10] Ricky T. Q. Chen, Brandon Amos, and Maximilian Nickel. Learning neural event functions for ordinary differential equations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. 3

[11] Katherine Crowson, Stefan Andreas Baumann, Alex Birch, Tanishq Mathew Abraham, Daniel Z Kaplan, and Enrico Shippole. Scalable high-resolution pixel-space image synthesis with hourglass diffusion transformers. *arXiv preprint arXiv:2401.11605*, 2024. 1

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 1, 2

[13] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In *Advances in Neural Information Processing Systems*, pages 8780–8794, 2021. 2

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020. 1, 3

[15] Samy Wu Fung, Howard Heaton, Qiuwei Li, Daniel McKenzie, Stanley J. Osher, and Wotao Yin. JFB: jacobian-free backpropagation for implicit networks. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 6648–6656. AAAI Press, 2022. 2, 3, 4, 5, 8

[16] Samir Yitzhak Gadre, Gabriel Ilharco, Alex Fang, Jonathan Hayase, Georgios Smyrnis, Thao Nguyen, Ryan Marten, Mitchell Wortsman, Dhruba Ghosh, Jieyu Zhang, Eyal Orgad, Rahim Entezari, Giannis Daras, Sarah M. Pratt, Vivek Ramanujan, Yonatan Bitton, Kalyani Marathe, Stephen Mussmann, Richard Vencu, Mehdi Cherti, Ranjay Krishna, Pang Wei Koh, Olga Saukh, Alexander Ratner, Shuran Song, Hannaneh Hajishirzi, Ali Farhadi, Romain Beaumont, Sewoong Oh, Alex Dimakis, Jenia Jitsev, Yair Carmon, Vaishaal Shankar, and Ludwig Schmidt. Datacomp: In search of the next generation of multimodal datasets. *CoRR*, abs/2304.14108, 2023. 1

[17] Zhengyang Geng and J. Zico Kolter. Torchdeq: A library for deep equilibrium models. https://github.com/locuslab/torchdeq, 2023. 2

[18] Zhengyang Geng, Xin-Yu Zhang, Shaojie Bai, Yisen Wang, and Zhouchen Lin. On training implicit models. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 24247–24260, 2021. 2

[19] Zhengyang Geng, Ashwini Pokle, and J Zico Kolter. One-step diffusion distillation via deep equilibrium models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 3

[20] Davis Gilton, Gregory Ongie, and Rebecca Willett. Deep equilibrium architectures for inverse problems in imaging. *IEEE Transactions on Computational Imaging*, 7:1123–1133, 2021. 2

[21] Andrzej Granas and James Dugundji. *Fixed point theory*. Springer, 2003. 3

[22] 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. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6626–6637, 2017. 5

[23] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, pages 6840–6851, 2020. 1, 2

[24] V.I. Istratescu. *Fixed Point Theory: An Introduction*. Springer Dordrecht, Dordrecht, 1 edition, 1981. eBook Packages Springer Book Archive. 3

[25] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *ArXiv*, abs/1710.10196, 2017. 1, 2

[26] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *arXiv preprint arXiv:2206.00364*, 2022. 6

[27] Patrick Kidger, James Morrill, James Foster, and Terry J. Lyons. Neural controlled differential equations for irregular time series. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. 3

[28] Patrick Kidger, James Foster, Xuechen Li, and Terry J. Lyons. Neural sdes as infinite-dimensional gans. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, pages 5453–5463. PMLR, 2021. 3

[29] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. 2, 3, 5

[30] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 1106–1114, 2012. 3

[31] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2019. 8

[32] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998. 3

[33] Shanchuan Lin, Bingchen Liu, Jiashi Li, and Xiao Yang. Common diffusion noise schedules and sample steps are flawed. *CoRR*, abs/2305.08891, 2023. 5

[34] Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. In *International Conference on Learning Representations*, 2021. 2

[35] 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. *arXiv preprint arXiv:2206.00927*, 2022. 6

[36] Alex 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. *arXiv preprint arXiv:2112.10741*, 2021. 2

[37] 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, 2022. 1

[38] William Peebles and Saining Xie. Scalable diffusion models with transformers. *CoRR*, abs/2212.09748, 2022. 1, 2, 5, 6

[39] Ashwini Pokle, Zhengyang Geng, and J Zico Kolter. Deep equilibrium approaches to diffusion models. *Advances in Neural Information Processing Systems*, 35:37975–37990, 2022. 2

[40] Machel Reid, Edison Marrese-Taylor, and Yutaka Matsuo. Subformer: Exploring weight sharing for parameter efficiency in generative transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4081–4090, 2021. 8

[41] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022. 1, 2, 3, 5

[42] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III*, 2015. 1, 2

[43] W. Rudin. *Principles of Mathematical Analysis*. McGraw-Hill, 3 edition, 1976. 5

[44] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2021. 5

[45] Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. LAION-400M: open dataset of clip-filtered 400 million image-text pairs. *CoRR*, abs/2111.02114, 2021. 1

[46] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. LAION-5B: an open large-scale dataset for training next generation image-text models. In *NeurIPS*, 2022. 1, 8

[47] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2020. 1, 2

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

[49] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *Proc. NeurIPS*, 33: 12438–12448, 2020. 2

[50] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. 2

[51] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. 1, 3

[52] J. Wallis. *A Treatise of Algebra, Both Historical and Practical*. John Playford, 1685. 3

[53] Daniel Watson, William Chan, Jonathan Ho, and Mohammad Norouzi. Learning fast samplers for diffusion models by differentiating through sample quality. In *International Conference on Learning Representations*, 2021. 2

[54] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *ArXiv*, abs/1506.03365, 2015. 1, 2

[55] Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. *arXiv preprint arXiv:2204.13902*, 2022. 6