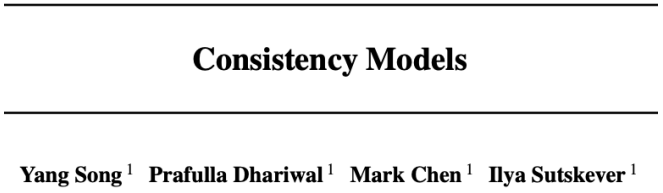
Consistency Models

Yang Song, Prafulla Dhariwal, Mark Chen, Ilya Sutskever

OpenAI

40th International Conference on Machine Learning, 2023



Introduction to Diffusion Models

- Diffusion models revolutionized the generative tasks (image, audio, video).
- Challenges: Iterative sampling limits real-time applications.
- Objectives: Develop efficient generative models with one-step generation.

$$\mathrm{d}\mathbf{x}_t = \boldsymbol{\mu}(\mathbf{x}_t, t)\,\mathrm{d}t + \sigma(t)\,\mathrm{d}\mathbf{w}_t, \qquad (1)$$

Equation 1 - SDE Formulation

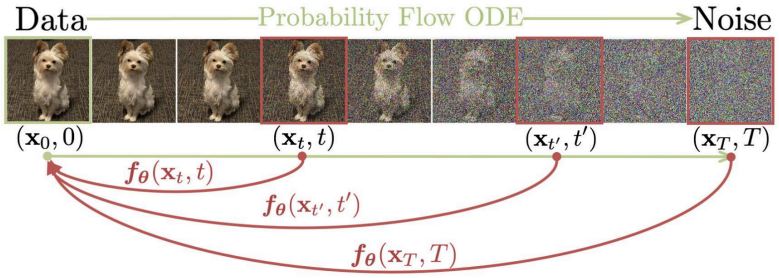


Figure 1: Given a Probability Flow (PF) ODE that smoothly converts data to noise, we learn to map any point (*e.g.*, $\mathbf{x}_t$, $\mathbf{x}_{t'}$, and $\mathbf{x}_T$) on the ODE trajectory to its origin (*e.g.*, $\mathbf{x}_0$) for generative modeling. Models of these mappings are called consistency models, as their outputs are trained to be consistent for points on the same trajectory.

Consistency Models Overview

- Maps points on ODE trajectories to their origins (self-consistency).
- Enables single-step sampling.
- Outperforms progressive distillation in sample quality.

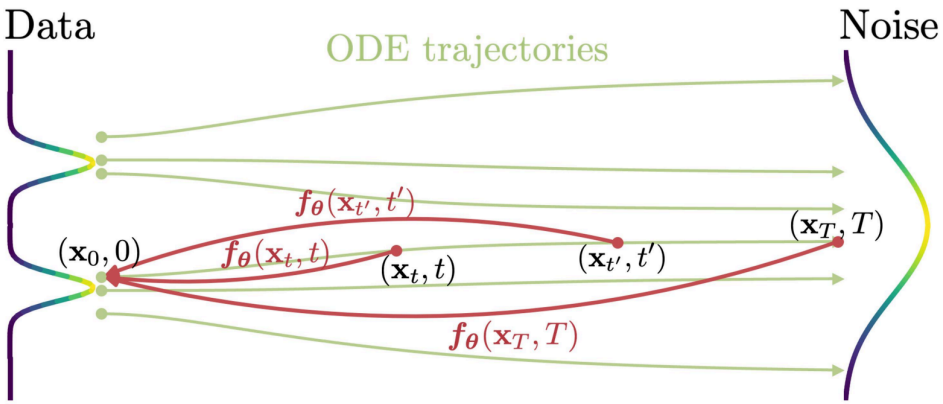


Figure 2: Consistency models are trained to map points on any trajectory of the PF ODE to the trajectory's origin.

Self-Consistency Property

- The **self-consistency property** in consistency models is a specific property in consistency models where all points (X_t) on the same trajectory in the Probability Flow ODE map to the same origin (X_ϵ), regardless of the time t or the intermediate pathways.
- **Consistency property** then refers generally to ensuring outputs align correctly with expectations, such as the mapping of a point on a trajectory to a consistent starting point (X_ϵ). And is achieved by enforcing **self-consistency.**
- In result:
  - This property enforces a fundamental **temporal consistency** between the model's outputs at different time steps.
  - Self-consistency ensures mappings are **consistent across all time steps** for points on the same trajectory.

Where is Self-Consistency applied?

- **Definition:** The consistency function $f(x_t, t)$ maps any point X_t at time t to the origin X_e, and the model satisfies:

  $f(x_t, t) = f(x_{t'}, t')$

  for any two time points t and t' along the same trajectory.
- **Enforcement:** The model is trained to satisfy this property using a consistency loss.
  - Training involves generating pairs of points x_t and x_t' from the same trajectory and minimizing the difference in their mapped outputs.

Where is Self-Consistency applied?

- **Loss Function**: The self-consistency is enforced during training by minimizing:

$$\mathcal{L}_{\text{consistency}} = \mathbb{E}\left[\|f(x_t, t) - f(x_{t'}, t')\|^2\right]$$

  This ensures that outputs from the model are consistent regardless of the path taken through the trajectory.
- **Parameterization**: The paper discusses parameterizing the consistency function to meet **boundary conditions** (identity at $t = \epsilon$ and enforce smooth transitions and gradients across time.
- **Finally** with these architectural choices the authors ensure the self-consistency property.

Mathematical Foundations in DMs

- Probability Flow ODE (PF ODE): Equation (2).
- Connection to SDE and sampling via Equation (3).
- Boundary conditions enforced for robustness.

$$\mathrm{d}\mathbf{x}_t = \left[\boldsymbol{\mu}(\mathbf{x}_t, t) - \frac{1}{2}\sigma(t)^2 \nabla \log p_t(\mathbf{x}_t)\right] \mathrm{d}t. \qquad (2)$$

Equation 2 - Probability Flow ODE

$$\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = -t\boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}_t, t). \qquad (3)$$

Equation 3 - the empirical PF ODE

Parameterization For any consistency function $\boldsymbol{f}(\cdot,\cdot)$, we have $\boldsymbol{f}(\mathbf{x}_\epsilon, \epsilon) = \mathbf{x}_\epsilon$, *i.e.*, $\boldsymbol{f}(\cdot, \epsilon)$ is an identity function. We call this constraint the *boundary condition*. All consistency models have to meet this boundary condition, as it plays a crucial role in the successful training of consistency models. This boundary condition is also the most confining architectural constraint on consistency models. For consistency models based on deep neural networks, we discuss two ways to implement this boundary condition *almost for free*. Suppose we have a free-form deep neural network $F_{\boldsymbol{\theta}}(\mathbf{x}, t)$ whose output has the same dimensionality as $\mathbf{x}$. The first way is to simply parameterize the consistency model as

$$\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}, t) = \begin{cases} \mathbf{x} & t = \epsilon \\ F_{\boldsymbol{\theta}}(\mathbf{x}, t) & t \in (\epsilon, T] \end{cases}. \qquad (4)$$

The second method is to parameterize the consistency model using skip connections, that is,

$$\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}, t) = c_{\text{skip}}(t)\mathbf{x} + c_{\text{out}}(t)F_{\boldsymbol{\theta}}(\mathbf{x}, t), \qquad (5)$$

where $c_{\text{skip}}(t)$ and $c_{\text{out}}(t)$ are differentiable functions such that $c_{\text{skip}}(\epsilon) = 1$, and $c_{\text{out}}(\epsilon) = 0$. This way, the consistency model is differentiable at $t = \epsilon$ if $F_{\boldsymbol{\theta}}(\mathbf{x}, t), c_{\text{skip}}(t), c_{\text{out}}(t)$ are all differentiable, which is critical for training continuous-time consistency models (Appendices B.1 and B.2). The parameterization in Eq. (5) bears

Mathematical Foundations [cont'd]

- The Probability Flow ODE (PF ODE) (Equation 2) is derived from the **stochastic differential equation (SDE)** (Equation 1) governing the diffusion process.
- The SDE (Equation 1) progressively adds noise to the data to transform it into a Gaussian distribution.

$$\mathrm{d}\mathbf{x}_t = \boldsymbol{\mu}(\mathbf{x}_t, t)\,\mathrm{d}t + \sigma(t)\,\mathrm{d}\mathbf{w}_t, \tag{1}$$

Equation 1 - Diffusion process SDE

$$\mathrm{d}\mathbf{x}_t = \left[\boldsymbol{\mu}(\mathbf{x}_t, t) - \frac{1}{2}\sigma(t)^2 \nabla \log p_t(\mathbf{x}_t)\right]\mathrm{d}t. \tag{2}$$

Equation 2 - Probability Flow ODE

$$\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = -t\boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}_t, t). \tag{3}$$

Equation 3 - the empirical PF ODE

Mathematical Foundations [cont'd]

- Then the PF ODE (Equation 2) describes the deterministic evolution of data under the same transformation as the diffusion SDE but without stochasticity. The key term, $-\frac{1}{2}\sigma^2(t)\nabla \log p_t(\mathbf{x}_t)$, is responsible for the noise-removal dynamics.

$$\mathrm{d}\mathbf{x}_t = \boldsymbol{\mu}(\mathbf{x}_t, t)\,\mathrm{d}t + \sigma(t)\,\mathrm{d}\mathbf{w}_t, \tag{1}$$

Equation 1 - Diffusion process SDE

$$\mathrm{d}\mathbf{x}_t = \left[\boldsymbol{\mu}(\mathbf{x}_t, t) - \frac{1}{2}\sigma(t)^2\nabla \log p_t(\mathbf{x}_t)\right]\mathrm{d}t. \tag{2}$$

Equation 2 - Probability Flow ODE

$$\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = -t\boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}_t, t). \tag{3}$$

Equation 3 - the empirical PF ODE

Mathematical Foundations [cont'd]

- And the equation 3 is a practical implementation of the PF ODE, using a pre-trained **score model** $s_\phi(X_t, t)$ to approximate the true score function $\nabla log p_t(X_t)$**.**
- $s_\phi(X_t, t) \approx \nabla log p_t(X_t)$ is a score-based model trained to predict the gradient of the log-density function at time t.
- The scalar t and and the negative sign are simplifications that arise from parameterizing $\sigma(t)$ and $\mu(x_t, t)$ in a specific way.

$$\mathrm{d}\mathbf{x}_t = \boldsymbol{\mu}(\mathbf{x}_t, t)\,\mathrm{d}t + \sigma(t)\,\mathrm{d}\mathbf{w}_t, \tag{1}$$

Equation 1 - Diffusion process SDE

$$\mathrm{d}\mathbf{x}_t = \left[\boldsymbol{\mu}(\mathbf{x}_t, t) - \frac{1}{2}\sigma(t)^2 \nabla \log p_t(\mathbf{x}_t)\right]\mathrm{d}t. \tag{2}$$

Equation 2 - Probability Flow ODE

$$\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = -t\boldsymbol{s}_{\boldsymbol{\phi}}(\mathbf{x}_t, t). \tag{3}$$

Equation 3 - the empirical PF ODE

Consistency Models Being Made

- **Core Idea:** Mapping Noise Directly to Data
- In diffusion models, we start with noise and iteratively denoise it step by step using a **score function** trained to estimate gradients of log probability densities.
- Consistency models instead learn a **consistency function** $f_\theta(x_t, t)$, which directly maps any noisy input x_t at time t to a consistent output x_ϵ (the clean data or low-noise version).

Consistency Models Being Made [cont'd]

- **Self-Consistency Property**
- Consistency models enforce the **self-consistency property:**

$$f_\theta(x_t, t) = f_\theta(f_\theta(x_t, t'), t')$$

- This ensures that the output remains consistent regardless of intermediate steps or the order in which t values are traversed.

Consistency Models Being Made [cont'd]

- **Training Consistency Models**
- Consistency models are trained using two approaches:
- Distillation from Diffusion Models:
  - Pre-trained diffusion models are used to generate pairs of adjacent points on the Probability Flow ODE trajectory.
  - A **consistency loss** is applied to minimize the difference between model outputs for these pairs.
- Direct Training (No Pre-Trained Models):
  - Instead of distilling from diffusion models, consistency models can be trained from scratch by using an unbiased estimator of the score function.

Training Consistency Models

- Two approaches: Distillation and isolation.
- Distillation: Leverages pre-trained diffusion models.
- Isolation: Trains consistency models from scratch.
- Loss function: Consistency distillation loss (Equation 7).

$$\mathcal{L}_{CD}^{N}(\boldsymbol{\theta}, \boldsymbol{\theta}^{-}; \boldsymbol{\phi}) := \mathbb{E}[\lambda(t_n) d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}^{-}}(\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}}, t_n))], \quad (7)$$

Equation 7 - Consistency Distillation Loss

$$\mathbb{E}[\lambda(t_n) d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x} + t_{n+1}\mathbf{z}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}^{-}}(\mathbf{x} + t_n\mathbf{z}, t_n))], \quad (10)$$

Equation 10 - Consistency Training Loss

$$\mathcal{L}_{CD}^{N}(\boldsymbol{\theta}, \boldsymbol{\theta}^{-}; \boldsymbol{\phi}) = \mathcal{L}_{CT}^{N}(\boldsymbol{\theta}, \boldsymbol{\theta}^{-}) + o(\Delta t), \quad (9)$$

Equation 9 - Relation of Consistency Distillation Loss and Consistency Training Loss

Training Consistency Models: Distillation

- Distillation relies on pre-trained diffusion models.
- Key idea: Enforce consistency between adjacent points on a PF ODE trajectory.

Algorithm 2 Consistency Distillation (CD)

Input: dataset $\mathcal{D}$, initial model parameter $\boldsymbol{\theta}$, learning rate η, ODE solver $\Phi(\cdot, \cdot; \boldsymbol{\phi})$, $d(\cdot, \cdot)$, $\lambda(\cdot)$, and μ

$\boldsymbol{\theta}^- \leftarrow \boldsymbol{\theta}$

repeat

Sample $\mathbf{x} \sim \mathcal{D}$ and $n \sim \mathcal{U}[\![1, N-1]\!]$

Sample $\mathbf{x}_{t_{n+1}} \sim \mathcal{N}(\mathbf{x}; t_{n+1}^2 \boldsymbol{I})$

$\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}} \leftarrow \mathbf{x}_{t_{n+1}} + (t_n - t_{n+1})\Phi(\mathbf{x}_{t_{n+1}}, t_{n+1}; \boldsymbol{\phi})$

$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi}) \leftarrow$

$\lambda(t_n) d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}}, t_n))$

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi})$

$\boldsymbol{\theta}^- \leftarrow \text{stopgrad}(\mu \boldsymbol{\theta}^- + (1-\mu)\boldsymbol{\theta})$

until convergence

Algorithm Overview (Algorithm 2):

1. Sample data point and its noisy version.
2. Compute adjacent points using an ODE solver.
3. Train model to minimize differences using the distillation loss (Equation 7).

Loss Analysis:

- Distillation loss ensures smooth mappings along trajectories.
- Includes EMA updates for stability.

Training Consistency Models: Isolation

- Isolation trains consistency models from scratch.
- Unbiased estimator replaces the need for pre-trained models.

Algorithm 3 Consistency Training (CT)

Input: dataset $\mathcal{D}$, initial model parameter $\boldsymbol{\theta}$, learning rate η, step schedule $N(\cdot)$, EMA decay rate schedule $\mu(\cdot)$, $d(\cdot,\cdot)$, and $\lambda(\cdot)$

$\boldsymbol{\theta}^- \leftarrow \boldsymbol{\theta}$ and $k \leftarrow 0$

repeat

Sample $\mathbf{x} \sim \mathcal{D}$, and $n \sim \mathcal{U}[\![1, N(k)-1]\!]$

Sample $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$

$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}^-) \leftarrow$

$\lambda(t_n) d(\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x} + t_{n+1}\mathbf{z}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x} + t_n \mathbf{z}, t_n))$

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}^-)$

$\boldsymbol{\theta}^- \leftarrow \operatorname{stopgrad}(\mu(k)\boldsymbol{\theta}^- + (1-\mu(k))\boldsymbol{\theta})$

$k \leftarrow k+1$

until convergence

Algorithm (Algorithm 3):

1. Sample data point and inject noise.

2. Train to minimize the consistency training (CT) loss (Equation 10).

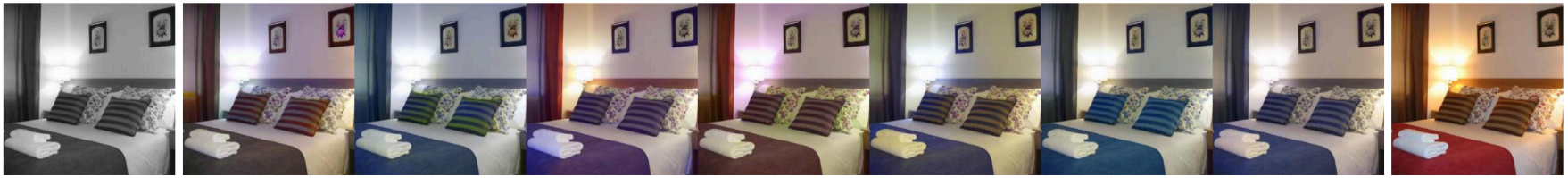
3. Progressively adjust parameters for improved convergence.

Loss Analysis:

- CT loss avoids pre-trained diffusion models.
- Adaptive schedules for step size and EMA improve performance.

Applications of Consistency Models

- Zero-shot data editing: Inpainting, colorization, super-resolution.
- Efficient trade-offs between compute and sample quality.



(a) *Left*: The gray-scale image. *Middle*: Colorized images. *Right*: The ground-truth image.

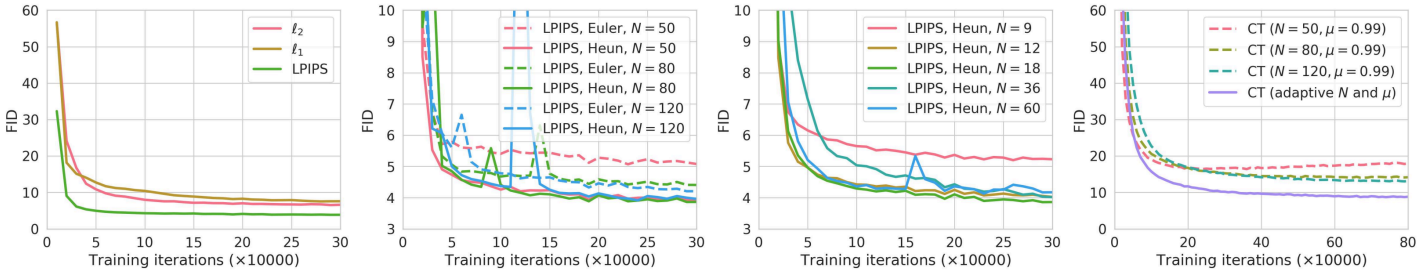
(b) *Left*: The downsampled image (32×32). *Middle*: Full resolution images (256×256). *Right*: The ground-truth image (256×256).

(c) *Left*: A stroke input provided by users. *Right*: Stroke-guided image generation.

Figure 6: Zero-shot image editing with a consistency model trained by consistency distillation on LSUN Bedroom 256×256.

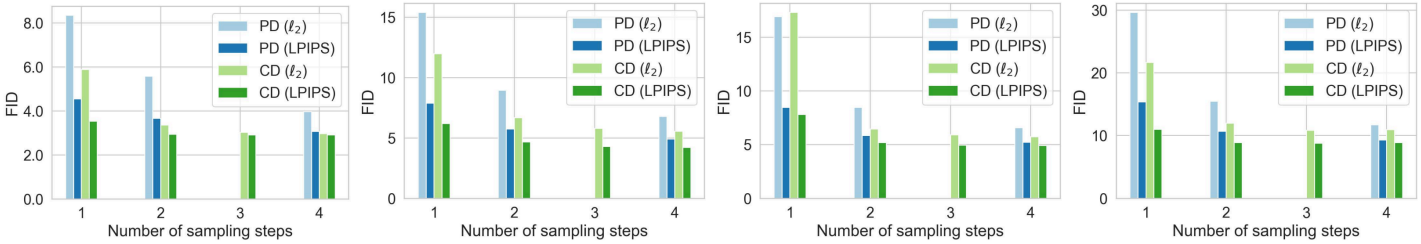
Experimental Results

- Benchmarks: CIFAR-10, ImageNet, LSUN.
- Metrics: FID, Inception Score.
- Comparison with state-of-the-art (Tables 1 & 2).



(a) Metric functions in CD. (b) Solvers and N in CD. (c) N with Heun solver in CD. (d) Adaptive N and μ in CT.

Figure 3: Various factors that affect consistency distillation (CD) and consistency training (CT) on CIFAR-10. The best configuration for CD is LPIPS, Heun ODE solver, and $N = 18$. Our adaptive schedule functions for N and μ make CT converge significantly faster than fixing them to be constants during the course of optimization.



(a) CIFAR-10 (b) ImageNet 64 × 64 (c) Bedroom 256 × 256 (d) Cat 256 × 256

Figure 4: Multistep image generation with consistency distillation (CD). CD outperforms progressive distillation (PD) across all datasets and sampling steps. The only exception is single-step generation on Bedroom 256 × 256.

Insights and Future Directions

- Theoretical insights: Convergence and estimation error (Theorem 1).
- Continuous vs. discrete training paradigms.
- Future work: Higher-order solvers, broader applications.

Conclusion

- Consistency models enable efficient single-step generation.
- Versatile for zero-shot editing tasks.

Q&A

- Thank you for your attention. I'm happy to take questions.