Consistency Models

Diffusion

- Forward: Diffuse $\mathbf{x}_0 \sim p_{\text{data}}(\mathbf{x})$ with a SDE

$$\mathrm{d}\mathbf{x} = \mathbf{f}\left(\mathbf{x}, t\right) \mathrm{d}t + g(t) \mathrm{d}\mathbf{w}$$

- Reverse: Denoise $\mathbf{x}_T \sim p_T(\mathbf{x})$ with the reverse-time SDE

$$\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right] \mathrm{d}t + g(t) \mathrm{d}\overline{\mathbf{w}}$$

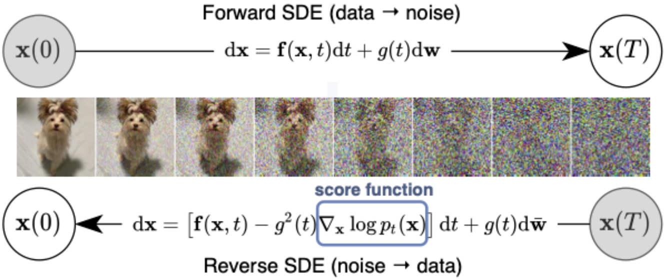

Figure 1: Diffusion model

Motivation

Iterative sampling: progressively denoising a random noise vector

+ Small-sized model can unroll into a larger computational graph: *Score model* $\mathbf{s}_{\phi}(\mathbf{x}_t, t)$ is typically UNet, where time embedding of t allows one model to deal with all the time step
– $\times$10 2000 sampling time compared to single-step generative models (e.g. GANs, VAEs, normalizing flows)

Can we make a **single-step generation** without sacrificing the advantage of iterative refinement?

Consistency Model (Overview)

- Use *Probability Flow (PF)* ODE instead of SDE to diffuse
- Learn model to have *self-consistency*: points on the same trajectory are mapped to the same initial point
- Use the model to retrieve the ODE trajectory initialized by a random noise vector

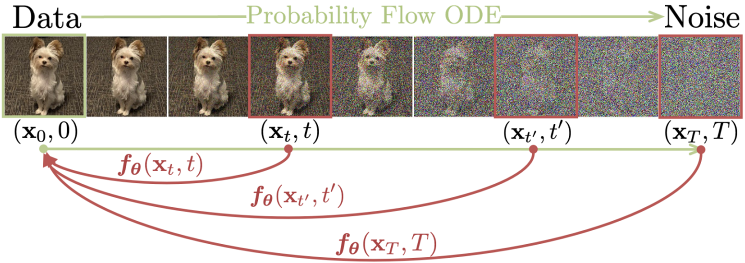

Figure 2: Consistency Model

Probability Flow ODE

Given a SDE

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

there exists a *Probability Flow* ODE, or a deterministic process

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

whose trajectory have the same marginal probability density as that of SDE.

Diffusion (PF ODE)

Sampling procedure of a diffusion model using PF ODE, with $\boldsymbol{\mu}(\mathbf{x}, t) = \mathbf{0}$, and $\sigma(t) = \sqrt{2t}$

1. Train a *score model* $\mathbf{s}_{\boldsymbol{\phi}}(\mathbf{x}, t) \simeq \nabla \log p_t(\mathbf{x})$ via score matching
2. Plug in $\mathbf{s}_{\boldsymbol{\phi}}$ in Eq. (1) to obtain the *empirical PF ODE*

$$\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = -t\mathbf{s}_{\boldsymbol{\phi}}(\mathbf{x}_t, t) \tag{2}$$

3. Sample $\hat{\mathbf{x}}_T \sim \pi = \mathcal{N}(\mathbf{0}, T^2\mathbf{I})$ to initialize the ODE
4. Solve the ODE with any numerical ODE solver to obtain a trajectory $\{\mathbf{x}_t\}_{t\in[0,T]}$
5. Then $\mathbf{x}_0$ can be view as an approximate of a sample from $p_{\text{data}}(\mathbf{x})$

* For numerical stability, one typically solve for $\mathbf{x}_\epsilon$ instead of $\mathbf{x}_0$

Consistency Models

Definition

Given a solution trajectory $\{\mathbf{x}_t\}_{t\in[\epsilon,T]}$ of a PF ODE in Eq. (1), the *consistency function* is a function defined by

$$\mathbf{f} : (\mathbf{x}_t, t) \mapsto \mathbf{x}_\epsilon.$$

A consistency function is *self-consistent*: if its outputs are consistent for any pairs $(\mathbf{x}_t, t)$ on the same trajectory.

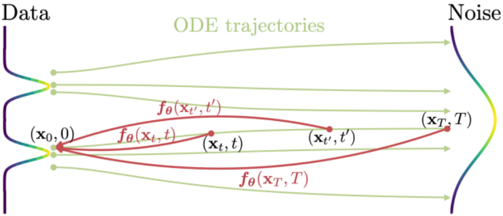

Figure 3: Consistency model and PF ODE trajectory

Consistency Models

Note for any consistency function $\mathbf{f}(\cdot, \cdot)$, $\mathbf{f}(\cdot, \epsilon)$ is an identity function. Hence to parameterize a consistency function, one must satisfy such *boundary condition*.

Parameterization (Simple)

We can simply parameterize a consistency function by

$$\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}, t) = \begin{cases} \mathbf{x} & t = \epsilon \\ F_{\boldsymbol{\theta}}(\mathbf{x}, t) & t \in (\epsilon, T] \end{cases} \tag{3}$$

Consistency Models

Parameterization (Skip Connection)

We can also parameterize a consistency function with a skip connection by

$$\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}, t) = c_{\text{skip}}(t)\mathbf{x} + c_{\text{out}}(t)F_{\boldsymbol{\theta}}(\mathbf{x}, t), \tag{4}$$

where c_{skip} and c_{out} are differentiable functions with $c_{\text{skip}}(\epsilon) = 1$, and $c_{\text{out}}(\epsilon) = 0$.

(4) has some advantage over (3):

1. Differentiable at $t = \epsilon$
2. Resemblance with strong diffusion architectures such as EDM.

Consistency Models

Training

1. Consistency Distillation (CD): Consistency model distills the knowledge of a pre-trained diffusion model into a single-step sampler
2. Consistency Training (CT): Consistency model is trained in isolation, without dependence on pre-trained diffusion models

Consistency Distillation

Given a pre-trained score model $\mathbf{s}_{\phi}(\mathbf{x}, t)$,

1. Discretize the time horizon $[\epsilon, T]$ into $N-1$ sub-intervals, with boundaries

$$\epsilon = t_1 < t_2 < \cdots < t_N = T.$$

2. Using a numerical solver, from $\mathbf{x}_{t_{n+1}}$ we can get an accurate approximate of $\mathbf{x}_{t_n}$ by

$$\hat{\mathbf{x}}_{t_n}^{\phi} := \mathbf{x}_{t_{n+1}} + (t_n - t_{n+1})\Phi(\mathbf{x}_{t_{n+1}}, t_{n+1}; \phi), \tag{5}$$

where $\Phi(\mathbf{x}, t; \phi)$ is the update function of a one-step ODE solver applied to empirical PF ODE

* For sufficiently large N, such approximation is accurate
* For example, using Euler solver, we have

$$\hat{\mathbf{x}}_{t_n}^{\phi} := \mathbf{x}_{t_{n+1}} - (t_n - t_{n+1})t_{n+1}\mathbf{s}_{\phi}(\mathbf{x}_{t_{n+1}}, t_{n+1})$$

Consistency Distillation

3. Sample $\mathbf{x} \sim p_{\text{data}}$, and randomly select t_n
4. Sample $\mathbf{x}_{t_{n+1}}$ from the transition density $\mathcal{N}(\mathbf{x}, t_{n+1}^2 \mathbf{I})$
5. Compute $\hat{\mathbf{x}}_{t_n}^{\phi}$ using ODE solver according to Eq. (5)
6. Train the consistency model by minimizing its output differences between $\mathbf{x}_{t_{n+1}}$ and $\hat{\mathbf{x}}_{t_n}^{\phi}$, using *consistency distillation loss* defined as

$$\mathcal{L}_{\text{CD}}^{N}(\boldsymbol{\theta}, \boldsymbol{\theta}^{-}; \boldsymbol{\phi}) := \mathbb{E}\left[\lambda(t_n) d\left(\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}), \mathbf{f}_{\boldsymbol{\theta}^{-}}(\hat{\mathbf{x}}_{t_n}^{\phi}, t_n)\right)\right] \quad (6)$$

- Expectation is taken over $\mathbf{x} \sim p_{\text{data}}, n \sim \mathcal{U}[1, N-1]$, and $\mathbf{x}_{t_{n+1}} \sim \mathcal{N}(\mathbf{x}, t_{n+1}\mathbf{I})$
- $\lambda(\cdot)$: positive weighting function
- $\boldsymbol{\theta}^{-}$: running average of past values of $\boldsymbol{\theta}$
- $d(\cdot, \cdot)$: metric function (e.g. $\ell_1 \ell_2$, LPIPS)

Consistency Distillation

Algorithm 1: 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

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

while *not converge* **do**

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

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

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

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

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

  $\boldsymbol{\theta}^- \leftarrow \texttt{stopgrad}\left(\mu \boldsymbol{\theta}^- + (1-\mu)\boldsymbol{\theta}\right)$

end

Consistency Distillation

Theorem 1 (Informal)

Under some reasonable regularity conditions, if $\mathcal{L}_{\text{CD}}^{N}(\boldsymbol{\theta}, \boldsymbol{\theta}; \boldsymbol{\phi}) = 0$, we have

$$\sup_{n,\mathbf{x}} \|\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}, t_n) - \mathbf{f}(\mathbf{x}, t_n; \boldsymbol{\phi})\|_2 = \mathcal{O}((\Delta t)^p).$$

Consistency Training

Need to estimate the score function $\nabla \log p_t(\mathbf{x}_t)$ *without* a pre-trained diffusion model. To do so, we use the following lemma:

Lemma 1

Let $\mathbf{x} \sim p_{\text{data}}(\mathbf{x}), \mathbf{x}_t \sim \mathcal{N}(\mathbf{x}_t; \mathbf{x}, t^2\mathbf{I})$ which results as

$$p_t(\mathbf{x}_t) = p_{\text{data}}(\mathbf{x}) \otimes \mathcal{N}(\mathbf{0}, t^2\mathbf{I}).$$

Then we have

$$\nabla \log p_t(\mathbf{x}) = -\mathbb{E}\left[\frac{\mathbf{x}_t - \mathbf{x}}{t^2} \,\middle|\, \mathbf{x}_t\right] \tag{7}$$

Consistency Training

Proof.

From $\nabla \log p_t(\mathbf{x}_t) = \nabla_{\mathbf{x}_t} \log \int p_{\mathsf{data}}(\mathbf{x}) p(\mathbf{x}_t \mid \mathbf{x}) \mathrm{d}\mathbf{x}$,

$$\begin{aligned}
\nabla \log p_t(\mathbf{x}_t) &= \frac{\int p_{\mathsf{data}}(\mathbf{x}) \nabla_{\mathbf{x}_t} p(\mathbf{x}_t \mid \mathbf{x}) \mathrm{d}\mathbf{x}}{\int p_{\mathsf{data}}(\mathbf{x}) p(\mathbf{x}_t \mid \mathbf{x}) \mathrm{d}\mathbf{x}} \\
&= \frac{\int p_{\mathsf{data}}(\mathbf{x}) p(\mathbf{x}_t \mid \mathbf{x}) \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t \mid \mathbf{x}) \mathrm{d}\mathbf{x}}{\int p_{\mathsf{data}}(\mathbf{x}) p(\mathbf{x}_t \mid \mathbf{x}) \mathrm{d}\mathbf{x}} \\
&= \frac{\int p_{\mathsf{data}}(\mathbf{x}) p(\mathbf{x}_t \mid \mathbf{x}) \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t \mid \mathbf{x}) \mathrm{d}\mathbf{x}}{p_t(\mathbf{x}_t)} \\
&= \int \frac{p_{\mathsf{data}}(\mathbf{x}) p(\mathbf{x}_t \mid \mathbf{x})}{p_t(\mathbf{x}_t)} \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t \mid \mathbf{x}) \mathrm{d}\mathbf{x} \\
&= \int p(\mathbf{x} \mid \mathbf{x}_t) \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t \mid \mathbf{x}) \mathrm{d}\mathbf{x} \\
&= \mathbb{E}\left[\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t \mid \mathbf{x}) \mid \mathbf{x}_t\right] \\
&= -\mathbb{E}\left[\frac{\mathbf{x}_t - \mathbf{x}}{t^2} \,\middle|\, \mathbf{x}_t\right]
\end{aligned}$$

Consistency Training

Theorem 2 (Informal)

Under some reasonable regularity conditions, if we use Euler ODE solver, we have

$$\mathcal{L}_{\mathsf{CD}}^{N}(\boldsymbol{\theta}, \boldsymbol{\theta}^{-}; \phi) = \mathcal{L}_{\mathsf{CT}}^{N}(\boldsymbol{\theta}, \boldsymbol{\theta}^{-}) + o(\Delta t), \tag{8}$$

where *consistency training* loss $\mathcal{L}_{\mathsf{CT}}^{N}(\boldsymbol{\theta}, \boldsymbol{\theta}^{-})$ is defined as

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

with $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

Consistency Training

Algorithm 2: Consistency Training (CT)

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

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

while *not converge* **do**

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

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

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

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

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

$k \leftarrow k + 1$

end

Consistency Models

Sampling (One-step)

Given a trained consistency model $\mathbf{f}_{\boldsymbol{\theta}}(\cdot,\cdot)$,

1. Sample $\hat{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{0}, T^2\mathbf{I})$
2. Evaluate $\hat{\mathbf{x}}_\epsilon = \mathbf{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}_T, T)$

Sampling (Multi-step)

Algorithm 3: Multi-Step Consistency Sampling

Input: Consistency model $\mathbf{f}_{\boldsymbol{\theta}}(\cdot,\cdot)$, sequence of time points $\tau_1 > \tau_2 > \cdots > \tau_{N-1}$, initial noise $\hat{\mathbf{x}}_T$

$\mathbf{x} \leftarrow \mathbf{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}_T, T)$

for $n = 1$ **to** $N-1$ **do**
- Sample $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- $\hat{\mathbf{x}}_{\tau_n} \leftarrow \mathbf{x} + \sqrt{\tau_n^2 - \epsilon^2}\mathbf{z}$
- $\mathbf{x} \leftarrow \mathbf{f}_{\boldsymbol{\theta}}\left(\hat{\mathbf{x}}_{\tau_n}, \tau_n\right)$

end

Experiments

METHOD	NFE (↓)	FID (↓)	IS (↑)
Diffusion + Samplers			
DDIM (Song et al., 2020)	50	4.67	
DDIM (Song et al., 2020)	20	6.84	
DDIM (Song et al., 2020)	10	8.23	
DPM-solver-2 (Lu et al., 2022)	12	5.28	
DPM-solver-3 (Lu et al., 2022)	12	6.03	
3-DEIS (Zhang & Chen, 2022)	10	**4.17**	
Diffusion + Distillation			
Knowledge Distillation* (Luhman & Luhman, 2021)	1	9.36	
DFNO* (Zheng et al., 2022)	1	4.12	
1-Rectified Flow (+distill)* (Liu et al., 2022)	1	6.18	9.08
2-Rectified Flow (+distill)* (Liu et al., 2022)	1	4.85	9.01
3-Rectified Flow (+distill)* (Liu et al., 2022)	1	5.21	8.79
PD (Salimans & Ho, 2022)	1	8.34	8.69
CD	1	**3.55**	**9.48**
PD (Salimans & Ho, 2022)	2	5.58	9.05
CD	2	**2.93**	**9.75**
Direct Generation			
BigGAN (Brock et al., 2019)	1	14.7	9.22
CR-GAN (Zhang et al., 2019)	1	14.6	8.40
AutoGAN (Gong et al., 2019)	1	12.4	8.55
E2GAN (Tian et al., 2020)	1	11.3	8.51
ViTGAN (Lee et al., 2021)	1	6.66	9.30
TransGAN (Jiang et al., 2021)	1	9.26	9.05
StyleGAN2-ADA (Karras et al., 2020)	1	2.92	**9.83**
StyleGAN-XL (Sauer et al., 2022)	1	**1.85**	
Score SDE (Song et al., 2021)	2000	2.20	**9.89**
DDPM (Ho et al., 2020)	1000	3.17	9.46
LSGM (Vahdat et al., 2021)	147	2.10	
PFGM (Xu et al., 2022)	110	2.35	9.68
EDM (Karras et al., 2022)	36	**2.04**	9.84
1-Rectified Flow (Liu et al., 2022)	1	378	1.13
Glow (Kingma & Dhariwal, 2018)	1	48.9	3.92
Residual Flow (Chen et al., 2019a)	1	46.4	
GLFlow (Xiao et al., 2019)	1	44.6	
DenseFlow (Grcić et al., 2021)	1	34.9	
DC-VAE (Parmar et al., 2021)	1	17.9	8.20
CT	1	**8.70**	**8.49**
CT	2	**5.83**	**8.85**

METHOD	NFE (↓)	FID (↓)	Prec. (↑)	Rec. (↑)
ImageNet 64 × 64				
PD† (Salimans & Ho, 2022)	1	15.39	0.59	0.62
DFNO†* (Zheng et al., 2022)	1	8.35		
CD†	1	6.20	0.68	0.63
PD† (Salimans & Ho, 2022)	2	8.95	0.63	**0.65**
CD†	2	**4.70**	**0.69**	0.64
ADM (Dhariwal & Nichol, 2021)	250	**2.07**	0.74	0.63
EDM (Karras et al., 2022)	79	2.44	0.71	**0.67**
BigGAN-deep (Brock et al., 2019)	1	4.06	**0.79**	0.48
CT	1	13.0	0.71	0.47
CT	2	11.1	0.69	0.56
LSUN Bedroom 256 × 256				
PD† (Salimans & Ho, 2022)	1	16.92	0.47	0.27
PD† (Salimans & Ho, 2022)	2	8.47	0.56	**0.39**
CD†	1	7.80	0.66	0.34
CD†	2	**5.22**	**0.68**	**0.39**
DDPM (Ho et al., 2020)	1000	4.89	0.60	0.45
ADM (Dhariwal & Nichol, 2021)	1000	**1.90**	0.66	**0.51**
EDM (Karras et al., 2022)	79	3.57	0.66	0.45
SS-GAN (Chen et al., 2019b)	1	13.3		
PGGAN (Karras et al., 2018)	1	8.34		
PG-SWGAN (Wu et al., 2019)	1	8.0		
StyleGAN2 (Karras et al., 2020)	1	2.35	0.59	0.48
CT	1	16.0	0.60	0.17
CT	2	7.85	**0.68**	0.33
LSUN Cat 256 × 256				
PD† (Salimans & Ho, 2022)	1	29.6	0.51	0.25
PD† (Salimans & Ho, 2022)	2	15.5	0.59	0.36
CD†	1	11.0	0.65	0.36
CD†	2	**8.84**	**0.66**	**0.40**
DDPM (Ho et al., 2020)	1000	17.1	0.53	0.48
ADM (Dhariwal & Nichol, 2021)	1000	**5.57**	0.63	**0.52**
EDM (Karras et al., 2022)	79	6.69	**0.70**	0.43
PGGAN (Karras et al., 2018)	1	37.5		
StyleGAN2 (Karras et al., 2020)	1	7.25	0.58	0.43
CT	1	20.7	0.56	0.23
CT	2	11.7	0.63	0.36

Figure 4: Sample quality on CIFAR-10 (left) ImageNet 64×64, LSUN Bedroom 256×256, Cat 256×256 (right)

Experiments

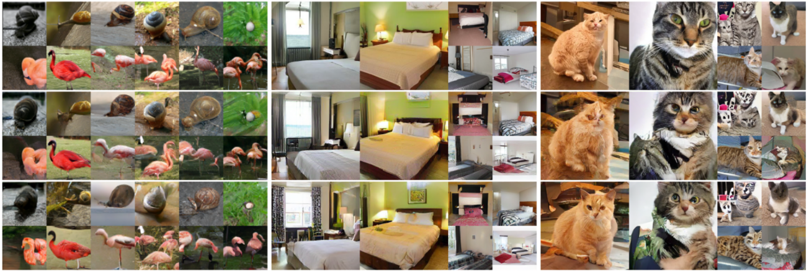

Figure 5: Sample generated by EDM (top), CT single-step (middle) CT 2-step (bottom)

Further Applications

1. Since consistency models define a one-to-one mapping between Gaussian noise and a data sample, one can interpolate between samples through latent space

Figure 6: Interpolating between images through latent space

Further Applications

2. As consistency models are trained to recover $\mathbf{x}_\epsilon$ from any noise input, they can perform denoising for various noise levels

Figure 7: Denoising various levels of noise from an image

Further Applications

3. Also using multi-step generation procedure, consistency models can solve various inverse problems (e.g. colorization, super-resolution, stroke-guided image generation) in zero-shot as diffusion models

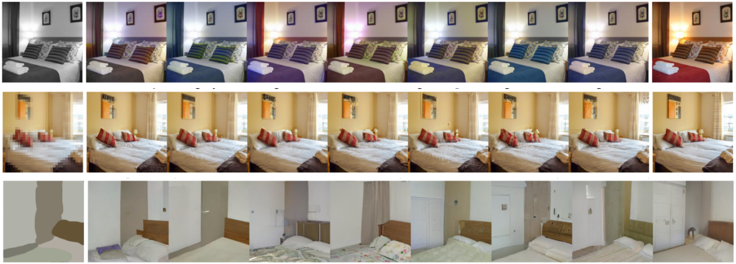

Figure 8: Colorization (top), super-resolution (middle), stroke-guided image generation (bottom)

Thank You

Q & A