Consistency Models

Pramook Khungurn

March 6, 2023

This note is written as I read "Consistency Models" by Song et al. [SDCS23].

1 Introduction

- Diffusion models are slow to sample, which means that they cannot be used in real-time applications like GANs, VAEs, and other models that can sample in one step.
- Diffusion models can be sped up by distillation: training "student" models to mimic the behavior of a "teacher" model. There are two existing techniques in literature.
  - Standard one-model-to-one-model distillation by Luhman and Luhman [LL21].
  - Progressive distillation by Salimans and Ho [SH22].
- The Consistenty Models paper proposes a new generative model that can generate high-quality sample fast.
- A consistency model can be trained in two ways.
  - As a way to distill an existing diffusion model.
  - From scratch, as a stand-alone model.
- Performance of consistency models depend on the way you train it.
  - When trained by distillation, it achieved SOTA FID on CIFAR-10 and ImageNet 64×64 when compared to other distilled diffusion models.
    - This method consistency outperforms progressive distillation.
  - When trained as a standalone model, it outperformed single-step non-adversarial models on CIFAR-10, ImageNet 64×64 and LSUN 256×256.
    - This method performs on par with progressive distillation.

  However, the method still loses to the best GANs in one-step generation.
- A consistency model has a number of other advantages.
  - With it, you can also sample in one step or multiple steps. Using multiple steps get you higher quality samples.
  - It also supports operations such as image-inpainting, colorization, and super-resolution.

2 Background

- Let $\mathbf{x}$ denote a data sample and $p_{\text{data}}(\mathbf{x})$ denote the probability distribution of the data.
- In a diffusion model, the data distribution p_{data} is corrupted with noise. Its evolution is governed by the stochastic differential equation (SDE)

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

  where

  - $t \in [0, T]$,
  - $\boldsymbol{\mu}(\mathbf{x}_t, t)$ is called the **drift coefficient**,
  - $\sigma(t)$ is called the **diffusion coefficient**, and
  - $\{\mathbf{w}_t\}_{t\in[0,T]}$ is the standard Brownian motion.

- Let $p_t(\mathbf{x})$ denote the distribution of $\mathbf{x}_t$.
- The boundary condition of the above SDE is $p_0(\mathbf{x}) = p_{\text{data}}(\mathbf{x})$.
- The above SDE has an ODE whose distribution of $\mathbf{x}_t$ (also denoted by $p_t(\mathbf{x})$) coincides with that of the SDE. This ODE is called the **probability flow ODE**. It is given by:

$$\mathrm{d}\mathbf{x}_t = \bigg(\boldsymbol{\mu}(\mathbf{x}_t, t) - \frac{1}{2}\sigma^2(t)\nabla_{\mathbf{x}_t}\log p_t(\mathbf{x}_t)\bigg)\,\mathrm{d}t$$

  where $\nabla_{\mathbf{x}_t}\log p_t(\mathbf{x}_t)$ is called the **score function** of $p_t(\mathbf{x}_t)$.

- The SDE is typically designed so that $p_T(\mathbf{x})$ is closed to a spherical Gaussian distribution $\pi(\mathbf{x})$.
- The paper adopts the formulation of Karras et al. [KAAL22] where

  - $\boldsymbol{\mu}(\mathbf{x}_t, t) = \mathbf{0}$, and
  - $\sigma(t) = \sqrt{2t}$.

  This gives

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

  where $*$ denotes the convolution operation.

- If we make T large enough so that $T^2 \gg \mathrm{Var}(\mathbf{x}_0)$. We have that $p_T(\mathbf{x}) \approx \pi(\mathbf{x}) = \mathcal{N}(\mathbf{0}, T^2 I)$.
- To train a diffusion model, we can train a network $\mathbf{s}_{\boldsymbol{\phi}}(\mathbf{x}, t)$, called the **score model**, to estimate $\nabla_{\mathbf{x}_t}\log p_t(\mathbf{x})_t$. The empirical estimate of the probability flow ODE then becomes

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

  which the paper calls the **empirical probability flow ODE**.

- To sample a data item, we start by sampling $\hat{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{0}, T^2 I)$. Then, we simulate the empirical probability flow ODE backward in time to $t = 0$ by any ODE solving method such as Euler [SSDK+21] and Huen's [KAAL22].
- To avoid numerical instability, we typically stop at $t = \epsilon$ where ϵ is a small positive constant and return $\hat{\mathbf{x}}_\epsilon$ as the output instead of $\hat{\mathbf{x}}_0$.
- The paper follows Karras et al. and uses $T = 80$ and $\epsilon = 0.002$. The pixel values are scaled to the range $[-1, 1]$.

3 Consistency Models

3.1 Definition

- Given a solution trajectory $\{\mathbf{x}_t\}_{t \in [\epsilon, T]}$ of the probability flow ODE, we define the **consistency function** $\mathbf{f} : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}$ as

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

- The consistency function has the following properties:
  - It is **self-consistent**, meaning that that $\mathbf{f}(\mathbf{x}_t, t) = \mathbf{f}(\mathbf{x}_{t'}, t')$ for all $t, t' \in [\epsilon, T]$.
  - With a fixed time argument, $\mathbf{f}(\cdot, t)$ is invertible.
- A **consistency model**, denoted by $\mathbf{f}_{\boldsymbol{\theta}}$, is trained to estimate the consistency function $\mathbf{f}$.

3.2 Parameterization

- For any consistency function, $\mathbf{f}(\cdot, \epsilon)$ is the identity function. This constraint is called the **boundary condition**.
- To create a function that respects the boundary condition, the paper chooses to parameterize the consistency model as:

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

  where
  - $c_{\text{skip}}(t)$ is a differentiable function such that $c_{\text{skip}}(\epsilon) = 1$,
  - $c_{\text{out}}(t)$ is a differentiable function such that $c_{\text{out}}(\epsilon) = 0$,
  - $\mathbf{F}_{\boldsymbol{\theta}}(\mathbf{x}, t)$ is a free-form neural network.
- The paper chooses

$$c_{\text{skip}}(t) = \frac{\sigma_{\text{data}}^2}{(t-\epsilon)^2 + \sigma_{\text{data}}^2},$$
$$c_{\text{out}}(t) = \frac{\sigma_{\text{data}}(t-\epsilon)}{\sqrt{\sigma_{\text{data}}^2 + t^2}}.$$

3.3 Sampling

- If we have a well-trained consistency model, we can generate a sample in one step by simply sampling $\hat{\mathbf{x}}_t \sim \mathcal{N}(\mathbf{0}, T^2 I)$ and then compute $\mathbf{f}_{\boldsymbol{\theta}}(\hat{\mathbf{x}}_T, T)$.
- Alternatively, we can sample in multiple steps. For this, we assume that we are given a sequence of time $\tau_1 > \tau_2 > \cdots > \tau_N$. Then, we may run the following algorithm.

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

end for

return $\mathbf{x}$.

- The paper find the times for the above algorithm with a greedy algorithm. The time is pinpointed one at a time using ternary search.
- Note that the above algorithm allows for many zero-shot data editing tasks such as inpainting, colorization, super-resolution, and SDEdit [MHS+21].

4 Training via Distillation

- First, we subdivide the time interval $[\epsilon, T]$ into $N-1$ intervals with $\epsilon = t_1 < t_2 < \cdots < t_{N-1} < t_N = T$.
- The paper follows Karras et al. and uses

$$t_i = \epsilon^{1/\rho} + \frac{i-1}{N-1}(T^{1/\rho} - \epsilon^{1/\rho})$$

with $\rho = 7$ [KAAL22].

- When N is sufficiently large, we can obtain accurate estimate of $\mathbf{x}_{t_n}$ from $\mathbf{x}_{t_{n+1}}$ by running one step of an ODE solver. With the Euler solver, this is given by

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

where $\mathbf{\Phi}(\cdot, \cdot; \boldsymbol{\phi})$ denotes the update performed by a one-step ODE solver.

- If we use the ODE

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

inspired by Karras et al., we have that the estimate is given by

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

- The examples used to train a consistency model is a tuple of the form $(\mathbf{x}^{\boldsymbol{\phi}}_{t_n}, \mathbf{x}_{t_{n+1}})$. Such an example can be generated by the following procedure:
  - Sampling $\mathbf{x}_0 \sim p_{\mathrm{data}}$.
  - Set $\mathbf{x}_{t_{n+1}} \leftarrow \mathbf{x}_0 + t_{n+1}\boldsymbol{\xi}$ where $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, I)$.
  - Compute $\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n} \leftarrow \mathbf{x}_{t_{n+1}} - (t_n - t_{n+1})t_{n+1}\mathbf{s}_{\boldsymbol{\phi}}(\mathbf{x}_{t_{n+1}}, t_{n+1})$.
- A consistency model is trained to minimize the output differences between $\mathbf{x}_{t_{n+1}}$ and $\mathbf{x}^{\boldsymbol{\phi}}_{t_n}$ according to the **consistency distillation loss**:

$$\mathcal{L}^N_{\mathrm{CD}}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi}) := E_{\substack{\mathbf{x}\sim p_{\mathrm{data}}, \\ n\sim\mathcal{U}(\{1,\dots,N-1\}), \\ \mathbf{x}_{n+1}\sim\mathcal{N}(\mathbf{x}, t^1_{n+1}I)}} \Big[\lambda(t_n) d(\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}), \mathbf{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}^{\boldsymbol{\phi}}_{t_n}, t_n))\Big].$$

Here,

  - $\mathcal{U}(\{1,\dots,N-1\})$ is the uniform distribution on $\{1,2,\dots,N-1\}$,
  - $\lambda(t_n)$ is the postivie weighting function,
  - $\boldsymbol{\theta}^-$ denotes a running average of the past values of $\boldsymbol{\theta}$ during the course of the optimization, and

- $d(\cdot, \cdot)$ is a metric function (that does not have to necessary satisfy the triangle inequality).

- For the metric function, the paper considered using
  - the ℓ_2 distance, $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2$,
  - the ℓ_1 distance, $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_1$, and
  - the LPIPS distance [ZIE+18].

- The paper found that $\lambda(t_n) = 1$ performs well across all datasets.

- Note that, while training, we deal with two separate networks.
  - $\mathbf{f}_{\boldsymbol{\theta}}$ is called the **online network**.
  - $\mathbf{f}_{\boldsymbol{\theta}^-}$ is caled the **target network**.

- The running average $\boldsymbol{\theta}^-$ is computed with exponential moving average. That is, given a decay rate $0 \leq \mu < 1$, we performed the following update.

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

- Here's the training algorithm.

  while not convergent **do**
  Sample $\mathbf{x} \sim p_{\text{data}}$.
  Sample $n \sim \mathcal{U}(\{1, 2, \dots, N-1\})$.
  Sample $\mathbf{x}_{t_{n+1}} \sim \mathcal{N}(\mathbf{x}, t_{n+1}^2 I)$.
  $\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}} \leftarrow \mathbf{x}_{t_{n+1}} + (t_n - t_{n+1})\boldsymbol{\Phi}(\mathbf{x}_{t_{n+1}}, t_{n+1}; \boldsymbol{\phi})$
  $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi}) \leftarrow \lambda(t_n) d(\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}_{t_{n+1}}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}^-}(\hat{\mathbf{x}}_{t_n}^{\boldsymbol{\phi}}, t_n))$
  Update $\boldsymbol{\theta}$ with $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}^-; \boldsymbol{\phi})$.
  $\boldsymbol{\theta}^- \leftarrow \mathrm{stopgrad}(\mu\boldsymbol{\theta}^- + (1-\mu)\boldsymbol{\theta})$
  end while

- The training procecure is similiar to momentum-based contrastive learning [GSA+20, HFW+19]. Using the running average can greatly stablize the training process and imporve the final performance of the consistency model.

- The paper shows that the following theorem is true.

  Theorem 1. *Let*
  - $\Delta t := \max_{n \in \{1, 2, \dots, N-1\}}\{t_{n+1} - t_n\}$, *and*
  - $\mathbf{f}(\cdot, \cdot; \boldsymbol{\phi})$ *be the consistency function of the empirical probability flow ODE* (2).

  Assume that $\mathbf{f}_{\boldsymbol{\theta}}$ *satisfies the Lipschitz condition:*

$$\|\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}, t) - \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{y}, t)\|_2 \leq L\|\mathbf{x} - \mathbf{y}\|_2$$

  for some positive constant L *and for all* $\mathbf{x}$, $\mathbf{y}$, *and* $t \in [\epsilon, T]$. *Assume further that, for all* $n \in \{1, 2, \dots, N-1\}$, *the ODE solver called at* t_{n+1} *has local error uniformly bounded by* $O((t_{n+1} - t_n)^{p_1})$ *with pgeq1. Then, if* $\mathcal{L}_{\text{CD}}^{\boldsymbol{\phi}}(\boldsymbol{\theta}, \boldsymbol{\theta}; \boldsymbol{\phi}) = 0$, *we have that*

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

- The consistency distillation loss can be extended to hold for infinitely many time steps. However, it requires Jacobian vector product and require forward-mode automatic differentiation for efficient implementation, which may not be well-supported in some deep learning frameworks.

5 Training in Isolation

- Consistency models can also be trained from scratch without a pre-trained diffusion model.
- When training with distillation (which we shall now refer to as "consistency distillation"), we need $\mathbf{s}_{\boldsymbol{\phi}}(\mathbf{x}, t)$ to approximate the score $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$. However, if we were to train a consistency model from scatch, we do not have this score network any more.
- However, because $\mathbf{x}_t \sim \mathcal{N}(\mathbf{x}_0, t^2 I)$, we have that, by Tweedie's formula,

  $$E[\mathbf{x}_0 | \mathbf{x}_t] = \mathbf{x}_t + t^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t).$$

  As a result,

  $$\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) = E\bigg[- \frac{\mathbf{x}_0 - \mathbf{x}_t}{t^2} \bigg| \mathbf{x}_t \bigg].$$

  In other words, we can use $-(\mathbf{x}_0 - \mathbf{x}_t)/t^2$ as an unbiased estimate of $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$. Here, $\mathbf{x}_0 \sim p_{\mathrm{data}}$ and $\mathbf{x}_t \sim \mathcal{N}(\mathbf{x}_0, t^2 I)$. This gives

  $$\begin{aligned} \hat{\mathbf{x}}_n^{\phi} &= \mathbf{x}_{t_{n+1}} - (t_n - t_{n+1}) t_{n+1} \frac{\mathbf{x}_0 - \mathbf{x}_{t_{n+1}}}{t_{n+1}^2} \\ &= \mathbf{x}_0 + t_{n+1}\boldsymbol{\xi} - (t_n - t_{n+1}) \frac{\mathbf{x}_0 - \mathbf{x}_0 - t_{n+1}\boldsymbol{\xi}}{t_{n+1}} \\ &= \mathbf{x}_0 + t_{n+1}\boldsymbol{\xi} + t_n \boldsymbol{\xi} - t_{n+1}\boldsymbol{\xi} \\ &= \mathbf{x}_0 + t_n \boldsymbol{\xi} \end{aligned}$$

  where $\boldsymbol{\xi}$ is a noise vector distributed according to $\mathcal{N}(\mathbf{0}, I)$ such that $\mathbf{x}_t = \mathbf{x}_0 + t_{n+1}\boldsymbol{\xi}$.
- With the above estimate, the consistency distillation loss becomes the **consistency training loss**:

  $$\mathcal{L}_{\mathrm{CT}}^N(\boldsymbol{\theta}, \boldsymbol{\theta}^-) = E_{\substack{\mathbf{x} \sim p_{\mathrm{data}}, \\ n \sim \mathcal{U}(\{1,2,\dots,N-1\}), \\ \boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, I)}} \Big[d\big(\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x} + t_{n+1}\boldsymbol{\xi}, t_{n+1}), f_{\boldsymbol{\theta}^-}(\mathbf{x} + t_n\boldsymbol{\xi}, t_n)\big) \Big].$$

- The training algorithm is given by

  while not convergent **do**

  Sample $\mathbf{x} \sim p_{\mathrm{data}}$.
  Sample $n \sim \mathcal{U}(\{1, 2, \dots, N-1\})$.
  Sample $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, I)$.
  $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}^-) \leftarrow \lambda(t_n) d(\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x} + t_{n+1}\boldsymbol{\xi}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}^-}(\mathbf{x} + t_n\boldsymbol{\xi}, t_n))$
  Update $\boldsymbol{\theta}$ with $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}^-)$.
  $\boldsymbol{\theta}^- \leftarrow \mathrm{stopgrad}(\mu\boldsymbol{\theta}^- + (1-\mu)\boldsymbol{\theta})$

  end while

- The following theorem characterize what the algorithm can achieve.

  Theorem 2. *Let* $\Delta t := \max_{n \in \{1,2,\dots,N-1\}} \{t_{n+1} - t_n\}$*. Assume that*

  - *the metric d and the target network $\mathbf{f}_{\boldsymbol{\theta}^-}$ are both twice continuously differentiable with bounded second derivatives,*
  - *the weighting function $\lambda(\cdot)$ is bounded,*
  - $E[\|\nabla_{\mathbf{x}} \log p_t(\mathbf{x})\|_2^2] < \infty$*, and*

  - *if we use the Euler ODE solver, the pre-trained score model maches the ground truth (i.e., $\mathbf{s}_{\boldsymbol{\phi}}(\mathbf{x}, t) = \nabla_{\mathbf{x}} p_t(\mathbf{x})$ for all $\mathbf{x}$ and t).*

Then,

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

Moreover, $\mathcal{L}_{\mathrm{CT}}^{N}(\boldsymbol{\theta}, \boldsymbol{\theta}^{-}) \geq O(\Delta t)$ if $\inf_N \mathcal{L}_{\mathrm{CD}}^{N}(\boldsymbol{\theta}, \boldsymbol{\theta}^{-}; \boldsymbol{\phi}) > 0$.

- From the above theorem, note that the loss $\mathcal{L}_{\mathrm{CT}}^{N}(\boldsymbol{\theta}, \boldsymbol{\theta}^{-}) \geq O(\Delta t)$ is greater than the remainder $\mathcal{L}_{\mathrm{CD}}^{N}(\boldsymbol{\theta}, \boldsymbol{\theta}^{-}; \boldsymbol{\phi}) - \mathcal{L}_{\mathrm{CT}}^{N}(\boldsymbol{\theta}, \boldsymbol{\theta}^{-}) = o(\Delta t)$, and the loss itself will dominate the as $N \to \infty$ and $\Delta t \to 0$.
- For improved performance, the paper proposes increasing N during training according to a schedule function $N(\cdot)$ that depends on the training iteration.
  - When N is small, it facilitates faster convergence at the beginning of training (i.e., less "variance"). However, the resulting model would have more "bias".
  - When N is a large, the model has more "variance" but less "bias". This is desirable at the end of training, where variance should have been suppressed in earlier phases.
  - As a result, N should increase as training progresses.
- The paper also found that μ should change along with N according to a schedule function $\mu(\cdot)$.
- The paper uses the following $N(\cdot)$ and $\mu(\cdot)$ functions.

$$\begin{aligned} N(k) &= \left\lceil \sqrt{\frac{k}{K}((s_1+1)^2 - s_0^2) + s_0^2 + 1} \right\rceil + 1 \\ \mu(k) &= \exp\left(\frac{s_0 \log \mu_0}{N(k)}\right) \end{aligned}$$

  where

  - k is the number of training iterations completed to far,
  - K is the total number of iterations,
  - s_0 is the intiial discretization steps,
  - $s_1 > s_0$ is the final discretization steps,
  - $\mu_0 > 0$ denotes the EMA decay rate at the beginning of model training.

  So, $N(k)$ increases and $\mu(k)$ decreases as training progresses.

- The consistency training loss can also be extended to the continous case where $N \to \infty$ and $\boldsymbol{\theta}^{-} = \mathrm{stopgrad}(\boldsymbol{\theta})$. However, like the consistency distillation loss, it requires forward-mode differentiation.

6 Experiments

- Datasets
  - CIFAR-10
  - ImageNet 64×64
  - LSUN Bedrom 256×256
  - LSUN Cat 256×256
- Metrics
  - FID
  - IS
  - Precision and recall [KKL+19]

6.1 Training Consistency Models

- The experiments in the section were performed to understand
  - the effect of the metric function d (ℓ_1, ℓ_2, and LPIPS),
  - the ODE solver in consistency distillation,
  - the effect of the number of discretization steps N in consistency distillation, and
  - the effect of the schedules function $N(\cdot)$ and $\mu(\cdot)$ in consistency training.
- The experiments were performed with only the CIFAR-10 dataset.
- Consistency distillation experiments.
  - The paper evaluated N values from the set $\{9, 12, 18, 36, 50, 60, 80, 129\}$.
  - The ODE solvers were the Euler solver and the Heun's 2nd order solver [KAAL22].
  - The consistency models have the same architecture as the pre-trained diffusion model and were initialized from the diffusion model.
- Consistency distillation results.
  - LPIPS outperformed both ℓ_1 and ℓ_2 by a large margin.
  - Heun ODE solver and $N = 18$ are the best choices, consistent with what Karras et al. recommended [KAAL22].
  - With the same N, Huen's solver uniformly outperforms the Euler solver.
- In the consistency training experiments, the models are initialized randomly.
- Consistency training results.
  - Convergence of training is highly sensitive to N. Smaller N leads to faster convergence bu worse sample. Higher N leads to slower convergence but better samples.
  - Adaptive schedules for N and μ significantly improves convergence speed and sample quality.

6.2 Few-Step image Generation

- The paper compared images generated by CT and CD with the following distillation baselines:
  - Progressive distillation (PD) [SH22],
  - Straight distillation [LL21], and
  - DFNO [ZNV$^+$22].

  Distilled models were trained from Karras et al.'s models [KAAL22].
- Observations with respective to PD and CD.
  - Using the LPIPS metrics improves PD results compared to the ℓ_2 distance in the original paper.
  - Both PD and CD improve as more sampling steps are used.
  - CD uniformly outperforms PD across all datasets, sampling steps, and metric functions, except one case.
- CD outperforms all the other two distillation approaches.
- CIFAR-10 dataset at one-step generation.

- The best model with regards to FID (1.85) is StyleGAN-XL [SSG22].
  - The best model with regards to IS (9.83) is StyleGAN2 with adaptive discrimination augmentation [KLA+19, KAH+20].
  - CD achieved FID of 3.55 and IS of 9.48, which numerically cannot beat the best models yet.
  - CT performed even worse than CT.

- For 1-step ImageNet 64×64 generation, CD and CT did not beat BigGAN-deep [BDS18] on any metrics.
- For 1-step LSUN Bedroom 256×256 generation, CD and CT still loses to StyleGAN2 on FID, but CD won on precision and recall, and CT won on precision ony.
- For 1-step LSUN Cat 256×256 generation, CD and CT both lose StyleGan2 on FID, but CD won on precision, but not recall.
- The metrics for CD and the GANs are quite close. I think this might be because consistency models do not have a way to trade diversity for fidelity like GANs yet?

6.3 Zero-Shot Image Editing

- The paper demonstrated that CD models can be used to colorize images, super-resolution images, do SDEdits, and inpaintings, interpolations, and denoising using the multi-step sampling algorithm.

References

[BDS18] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis, 2018.

[GSA+20] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, koray kavukcuoglu, Remi Munos, and Michal Valko. Bootstrap your own latent - a new approach to self-supervised learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21271–21284. Curran Associates, Inc., 2020.

[HFW+19] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning, 2019.

[KAAL22] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models, 2022.

[KAH+20] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. In *Proc. NeurIPS*, 2020.

[KKL+19] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models, 2019.

[KLA+19] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan, 2019.

[LL21] Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed, 2021.

[MHS+21] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations, 2021.

[SDCS23] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models, 2023.

[SH22] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *CoRR*, abs/2202.00512, 2022.

[SSDK+21] 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.

[SSG22] Axel Sauer, Katja Schwarz, and Andreas Geiger. Stylegan-xl: Scaling stylegan to large diverse datasets, 2022.

[ZIE+18] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric, 2018.

[ZNV+22] Hongkai Zheng, Weili Nie, Arash Vahdat, Kamyar Azizzadenesheli, and Anima Anandkumar. Fast sampling of diffusion models via operator learning, 2022.