ACT: Adversarial Consistency Models

Fei Kong[1] Jinhao Duan[2] Lichao Sun[3] Hao Cheng[4] Renjing Xu[4]
Hengtao Shen[1] Xiaofeng Zhu[1] Xiaoshuang Shi[1]* Kaidi Xu[2]*
[1]University of Electronic Science and Technology of China
[2]Drexel University
[3]Lehigh University
[4]The Hong Kong University of Science and Technology (Guangzhou)
kong13661@outlook.com xsshi2013@gmail.com kx46@drexel.edu

Abstract

Though diffusion models excel in image generation, their step-by-step denoising leads to slow generation speeds. Consistency training addresses this issue with single-step sampling but often produces lower-quality generations and requires high training costs. In this paper, we show that optimizing consistency training loss minimizes the Wasserstein distance between target and generated distributions. As timestep increases, the upper bound accumulates previous consistency training losses. Therefore, larger batch sizes are needed to reduce both current and accumulated losses. We propose Adversarial Consistency Training (ACT), which directly minimizes the Jensen-Shannon (JS) divergence between distributions at each timestep using a discriminator. Theoretically, ACT enhances generation quality, and convergence. By incorporating a discriminator into the consistency training framework, our method achieves improved FID scores on CIFAR10 and ImageNet 64×64, retains zero-shot image inpainting capabilities, and uses less than $1/6$ of the original batch size and fewer than $1/2$ of the model parameters and training steps compared to the baseline method, this leads to a substantial reduction in resource consumption.

1. Introduction

Diffusion models, known for their success in image generation [11, 18, 39, 40, 48], utilize diffusion processes to produce high-quality, diverse images. They also perform tasks like zero-shot inpainting [28] and audio generation [22, 23, 32]. However, they have a significant drawback: lengthy sampling times. These models generate target distribution samples by iterative denoising a Gaussian noise input, a process that involves gradual noise reduction until samples match the target distribution. This limitation affects their practicality and efficiency in real-world applications.

The lengthy sampling times of diffusion models have spurred the creation of various strategies to tackle this issue. Several models and techniques have been suggested to enhance the efficiency of diffusion-based image generation [4, 26, 52]. Recently, consistency models [41] have been introduced to speed up the diffusion models' sampling process. A consistency function is one that consistently yields the same output along a specific trajectory. To use consistency models, the trajectory from noise to the target sample must be obtained. By fitting the consistency function, the model can generate data within 1 or 2 steps.

The score-based model [40], an extension of the diffusion model in continuous time, gradually samples from a normal distribution p_T to the sample distribution p_0. In deterministic sampling, it essentially solves an Ordinary Differential Equation (ODE), with each sample representing an ODE trajectory. Consistency models generate samples using a consistency function that aligns every point on the ODE trajectory with the ODE endpoint. However, deriving the true ODE trajectory is complex. To tackle this, consistency models suggest two methods. The first, consistency distillation, trains a score-based model to obtain the ODE trajectory. The second, consistency training, approximates the trajectory using a conditional one. Compared to distillation, consistency training has a larger error, leading to lower sample quality. The consistency function is trained by equating the model's output at time t_{n+1} with its output at time t_n.

Generative Adversarial Networks (GANs) [3, 14, 50], unlike consistency training, can directly minimize the distance between the model's generated and target distributions via the discriminator, independent of the model's output at previous time t_{n-1}. Drawing from GANs, we introduce Adversarial Consistency Training. We first theoretically explain the need for large batch sizes in consistency training

*Equal corresponding author

by showing its equivalence to optimizing the upper bound of the Wasserstein-distance between the model's generated and target distributions. This upper bound consists of the accumulated consistency training loss $\mathcal{L}_{CT}^{t_k}$, the distance between sampling distributions, and the accumulated error, all of which increase with t. Hence, a large batch size is crucial to minimize the error from the previous time t. To mitigate the impact of $\mathcal{L}_{CT}^{t_k}$ and accumulated error, we incorporate the discriminator into consistency training, enabling direct reduction of the JS-divergence between the generated and target distributions at each timestep t. Our experiments on CIFAR10 and ImageNet 64×64 show that ACT significantly surpasses consistency training while needing less than $1/6$ of the original batch size and less than $1/2$ of the original model parameters and training steps, leading to considerable resource savings. For comparison, we use 1 NVIDIA GeForce RTX 3090 for CIFAR10 and 4 NVIDIA A100 GPUs for ImageNet 64×64, while consistency training requires 8 and 64 A100 GPUs for CIFAR10 and ImageNet 64×64, respectively.

Our contributions are summarized as follows:

- We demonstrate that consistency training is equivalent to optimizing the upper bound of the W-distance. By analyzing this upper bound, we have identified one reason why consistency training requires a larger batch size.
- Following our analysis, we propose Adversarial Consistency Training (ACT) to directly optimize the JS divergence between the sampling distribution and the target distribution at each timestep t, by incorporating a discriminator into the consistency training process.
- Experimental results demonstrate that the proposed ACT significantly outperforms the original consistency models with only less than $1/6$ of the original batch size and less than $1/2$ of the training steps. This leads to a substantial reduction in resource consumption.

2. Related works

Generative Adversarial Networks GANs have achieved tremendous success in various domains, including image generation [14, 47, 49] and audio synthesis [9]. However, GAN training faces challenges such as instability and mode collapse, where the generator fails to capture the diversity of the training data. To address these issues, several methods have been proposed. For example, spectral normalization, gradient penalty, and differentiable data augmentation techniques have been developed. Spectral normalization [29] constrains the Lipschitz constant of the discriminator, promoting more stable training. Gradient penalty, as employed in the WGAN-GP [16], utilizes the gradient penalty to discriminator to limit the range of gradient, so as to avoid the tend of concentrating the weights around extreme values, when using weight clipping in WGAN [1]. [44] introduces the concept of zero centered gradient penalty, and StyleGAN2 [43] introduces lazy regularization which performs multiple steps of iteration before computing the gradient penalty to improve the efficiency. Moreover, differentiable data augmentation techniques [51] have been introduced to enhance the diversity and robustness of GAN models during training. StyleGAN2-ADA [42] improves GAN performance on small datasets by employing adaptive differentiable data augmentation techniques.

Diffusion Models Diffusion models have emerged as highly successful approaches for generating images [33, 34]. In contrast to the traditional approach of Generative Adversarial Networks (GANs), which involve a generator and a discriminator, diffusion models generate samples by modeling the inverse process of a diffusion process from Gaussian noise. Diffusion models have shown superior stable training process compared to GANs, effectively addressing issues such as checkerboard artifacts [10, 12, 36]. The diffusion process is defined as follows: $\boldsymbol{x}_t = \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{\beta_t}\boldsymbol{\epsilon}_t, \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. As t increases, β_t gradually increases, causing $\boldsymbol{x}_t$ to approximate random Gaussian noise. In the reverse diffusion process, $\boldsymbol{x}_t'$ follows a Gaussian distribution, assuming the same variance as in the forward diffusion process. The mean of $\boldsymbol{x}_t'$ is defined as: $\tilde{\boldsymbol{\mu}}_t = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\bar{\boldsymbol{\epsilon}}_\theta(\boldsymbol{x}_t, t)\right)$, where $\bar{\alpha}_t = \prod_{k=0}^{t}\alpha_k$ and $\bar{\alpha}_t + \bar{\beta}_t = 1$. The reverse diffusion process becomes: $\boldsymbol{x}_{t-1} = \tilde{\boldsymbol{\mu}}_t + \sqrt{\beta_t}\boldsymbol{\epsilon}, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The loss function is defined as $\mathbb{E}_{x_0, \bar{\boldsymbol{\epsilon}}_t}\left[\left\|\bar{\boldsymbol{\epsilon}}_t - \boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\bar{\boldsymbol{\epsilon}}_t, t\right)\right\|^2\right]$. Score-based models [40] transforms the discrete-time diffusion process into a continuous-time process and employs Stochastic Differential Equations (SDEs) to express the diffusion process. Moreover, the forward and backward processes are no longer restricted to the diffusion process. They employ the forward process defined as $d\boldsymbol{x} = \left(\boldsymbol{f}_t(\boldsymbol{x}) - \frac{1}{2}\left(g_t^2 - \sigma_t^2\right)\nabla_{\boldsymbol{x}}\log p_t(\boldsymbol{x})\right)dt + \sigma_t d\boldsymbol{w}$, and the corresponding backward process is $d\boldsymbol{x} = \left(\boldsymbol{f}_t(\boldsymbol{x}) - \frac{1}{2}\left(g_t^2 + \sigma_t^2\right)\nabla_{\boldsymbol{x}}\log p_t(\boldsymbol{x})\right)dt + \sigma_t d\bar{\boldsymbol{w}}$, where $\boldsymbol{w}$ is the forward time Brownian motion and $\bar{\boldsymbol{w}}$ is the forward time Brownian motion. Compared to GANs, diffusion models have longer sampling time consummations. Several methods have been proposed to accelerate the generation process, including [8, 35, 46], DDIM [38], Consistency models [41], etc.

Consistency type models A function is called a consistency function if its output is the same at every point on a trajectory. Formally, given a trajectory, $\boldsymbol{x}_t, t \in [0, T]$, the function satisfies $f(\boldsymbol{x}_{t_1}) = \mathbb{E}[f(\boldsymbol{x}_{t_2})]$, if $t_1, t_2 \in [0, T]$. If this trajectory is not a probability trajectory, then the expected symbol $\mathbb{E}$ in the above formula can be removed. [6] proposes Consistency Diffusion Models (CDM), which proves that when the forward diffusion process satisfies $d\boldsymbol{x}_t = g(t)d\boldsymbol{w}_t$, $\boldsymbol{h}(\boldsymbol{x}, t) = \nabla\log q_t(\boldsymbol{x})g^2(t) + \boldsymbol{x}$ is a consistency function. They add consistency regularity above during training to improve the sampling effectiveness of the model. [41] proposed consistency models. Unlike consistency diffusion models, Consistency Models (CM) utilize deterministic sampling to obtain a one-step sampling model by learning the mapping from each point $\boldsymbol{x}_t$ on the trajectory to $\boldsymbol{x}_0$. When training a diffusion model to obtain the trajectory $\boldsymbol{x}_t$, it is called consistency distillation. When using conditional-trajectories to approximate non-conditional trajectories, it is called consistency training. Compared to consistency distillation, consistency training has a lower sampling effectiveness.

3. Method

3.1. Preliminary

3.1.1 Score-Based Generative Models

Score-Based Generative Models [40], as an extension of diffusion models, extends the diffusion to continuous time, and the forward and backward processes are no longer limited to the diffusion process. Given a distribution p_t, where $t \in [0,T]$, p_0 is the data distribution and p_T is normal distribution. From p_0 to p_T, this distribution increasingly approximates a normal distribution. We sample $\boldsymbol{x}_t$ from p_t distribution. If we can obtain $\boldsymbol{x}_{t'}$ from the formula $d\boldsymbol{x} = \left(\boldsymbol{f}_t(\boldsymbol{x}) - \frac{1}{2}\left(g_t^2 - \sigma_t^2\right)\nabla_{\boldsymbol{x}}\log p_t(\boldsymbol{x})\right)dt + \sigma_t d\boldsymbol{w}$, where $\boldsymbol{w}$ is the forward time Brownian motion and $t' > t$, then we can obtain $\boldsymbol{x}_{t'}$ from the formula $d\boldsymbol{x} = \left(\boldsymbol{f}_t(\boldsymbol{x}) - \frac{1}{2}\left(g_t^2 + \sigma_t^2\right)\nabla_{\boldsymbol{x}}\log p_t(\boldsymbol{x})\right)dt + \sigma_t d\boldsymbol{w}$, where $\boldsymbol{w}$ is the backward time Brownian motion and $t' < t$. If $\sigma_t = 0$, this formula turns into a ordinary differential equation $d\boldsymbol{x} = \left(\boldsymbol{f}_t(\boldsymbol{x}) - \frac{1}{2}g_t^2\nabla_{\boldsymbol{x}}\log p_t(\boldsymbol{x})\right)dt$. We can generate a new sample by numerically solving this Ordinary Differential Equation (ODE). For each $\boldsymbol{x}_T \sim p_T$, this ODE describes a trajectory from $\boldsymbol{x}_T$ to $\boldsymbol{x}_0$.

3.1.2 Consistency Training

Denote $\{\boldsymbol{x}_t\}$ as a ODE trajectory, a function is called consistency function, if $\boldsymbol{g}(\boldsymbol{x}_{t_1}, t_1) = \boldsymbol{g}(\boldsymbol{x}_{t_2}, t_2)$, for any $\boldsymbol{x}_{t_1}, \boldsymbol{x}_{t_2} \in \{\boldsymbol{x}_t\}$. To reduce the time consumption for sampling from diffusion models, consistency training utilizes a model to fit the consistency function $\boldsymbol{g}(\boldsymbol{x}_{t_1}, t_1) = \boldsymbol{g}(\boldsymbol{x}_{t_2}, t_2) = \boldsymbol{x}_0$. The ODE trajectory selected by consistency training is

$$d\boldsymbol{x} = t\nabla_{\boldsymbol{x}}\log p_t(\boldsymbol{x})dt, t \in [0,T]. \tag{1}$$

In this setting, the distribution of

$$p_t(\boldsymbol{x}) = p_0(\boldsymbol{x}) * \mathcal{N}(0, t^2\boldsymbol{I}),$$

where $*$ is convolution operator. The consistency models are denoted as $\boldsymbol{f}(\boldsymbol{x}_t, t, \boldsymbol{\theta})$. Consistency model is defined as

$$\boldsymbol{f}(\boldsymbol{x}_t, t, \boldsymbol{\theta}) = \frac{0.5^2}{r_t^2 + 0.5^2}\boldsymbol{x}_t + \frac{0.5r_t}{\sqrt{0.5^2 + r_t^2}}\boldsymbol{F}_{\boldsymbol{\theta}}((\frac{1}{\sqrt{r_t^2 + 0.5^2}})\boldsymbol{x}_t, t), \tag{2}$$

where $\boldsymbol{\theta}$ represents the parameters of the model, $\boldsymbol{F}_{\boldsymbol{\theta}}$ is the output of network, $r_t = t - \epsilon$, and ϵ is a small number for numeric stability.

To train the consistency model $\boldsymbol{f}(\boldsymbol{x}_t, t, \theta)$, we need to divide the time interval $[0,T]$ into several discrete time steps, denoted as $t_0 = \epsilon < t_1 < t_2 < \cdots < t_N = T$. N gradually increases as the training progresses, satisfying

$$N(k) = \lceil\sqrt{\frac{k}{K}((s_1+1)^2 - s_0^2) + s_0^2} - 1\rceil + 1,$$

where K denotes the total number of training steps, s_1 is the end of time steps, s_0 is the beginning of time steps and k refers to the current training step. Denote

$$\mathcal{L}_{CD}^n = \sum_{k=1}^{n}\mathbb{E}[d(\boldsymbol{f}(\boldsymbol{x}_{t_k}, t_k, \boldsymbol{\theta}), \boldsymbol{f}(\boldsymbol{x}_{t_{k-1}}^{\Phi}, t_{k-1}, \boldsymbol{\theta}^-))],$$

where $d(\cdot)$ is a distance function, $\boldsymbol{\theta}^-$ is the exponentially moving average of each batch of $\boldsymbol{\theta}$, and $\boldsymbol{x}_{t_{n+1}} \sim p_{t_{n+1}}$. $\boldsymbol{x}_{t_n}^{\Phi}$ is obtained from $\boldsymbol{x}_{t_{n+1}}$ through the ODE solver Φ using Eq. (1). About $\boldsymbol{\theta}$ and $\boldsymbol{\theta}^-$, the equation is given as $\boldsymbol{\theta}_{k+1}^- = \mu(k)\boldsymbol{\theta}_k^- + (1-\mu(k))\boldsymbol{\theta}_k$, where $\mu(k) = \exp(\frac{s_0\log\mu_0}{N(k)})$ and μ_0 is the coefficient at the beginning.

However, calculating $\mathcal{L}_{CD}^{\Phi}$ requires training another score-based generative model. They also propose using conditional trajectories to approximate $x_{t_n}^{\Phi}$. This loss is denoted as

$$\mathcal{L}_{CT}^n = \sum_{k=1}^{n}\mathbb{E}[d(f(\boldsymbol{x}_0 + t_k\boldsymbol{z}, t_k, \boldsymbol{\theta}), f(\boldsymbol{x}_0 + t_{k-1}\boldsymbol{z}, t_{k-1}, \boldsymbol{\theta}^-))],$$

where $\boldsymbol{x}_0 \sim p_0$ and $\boldsymbol{z} \sim \mathcal{N}(0, I)$. $\mathcal{L}_{CT} = \mathcal{L}_{CT}^N$ is called consistency training loss. Using this loss to train the consistency model is called consistency training. This loss is proven [41] to satisfy

$$\mathcal{L}_{CT}^n = \mathcal{L}_{CD}^n + o(\Delta t), \tag{3}$$

when the ODE solver Φ is Euler solver.

3.1.3 Generative Adversarial Networks

Generative Adversarial Networks (GANs), as generative models, are divided into two parts during training. One part is the generator, denoted as $G(\cdot)$, which is used to generate samples from the approximated target distribution. The other part is the discriminator, denoted as $D(\cdot)$. The training of GANs is alternatively optimizing $G(\cdot)$ and $D(\cdot)$: 1) train to distinguish whether the sample is a generated sample; 2) train $G(\cdot)$ to deceive the discriminator. These two steps are alternated in training. One type of GANs can be described as the following minimax problem: $\min_G \max_D V(G,D) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$. It can be proven that this minimax problem is equivalent to minimizing the JS-divergence between p_{data} and $G(\boldsymbol{z})$, where $\boldsymbol{z} \sim p_{\boldsymbol{z}}$.

To improve the training stability of GANs, many methods have been proposed. A practical approach is the zero-centered gradient penalty. This is achieved by using the following regularization:

$$\mathcal{L}_{gp} = \|\nabla_{\boldsymbol{x}} D(\boldsymbol{x})\|^2, \boldsymbol{x} \sim p_{\text{data}}. \tag{4}$$

To reduce computational overhead, this regularization can be applied intermittently every few training steps, rather than at every step.

3.2. Analysis the Loss Function

Theorem 3.1. *If the consistency model satisfies the Lipschitz condition: there exists $L > 0$ such that for all $\boldsymbol{x}$, $\boldsymbol{y}$ and t, we have $\|\boldsymbol{f}(\boldsymbol{x}, t, \boldsymbol{\theta}) - \boldsymbol{f}(\boldsymbol{y}, t, \boldsymbol{\theta})\|_2 \leq L\|\boldsymbol{x} - \boldsymbol{y}\|_2$, then minimizing the consistency loss will reduce the upper boundary of the W-distance between the two distributions. This can be formally articulated as the following theorem:*

$$\begin{aligned} \mathcal{W}[f_{t_k}, g_{t_k}] &= \mathcal{W}[f_{t_k}, p_0] \\ &\leq L\mathcal{W}[q_{t_k}, p_{t_k}] + \mathcal{L}_{CT}^{t_k} + t_k O(\Delta t) + o(\Delta t), \end{aligned} \tag{5}$$

where the definition of p_t, $\boldsymbol{f}$, $\mathcal{L}_{CT}^{t_k}$ and $\boldsymbol{g}$ is consistent with that in Sec. 3.1.2. $\Delta t = \max(t_k - t_{k-1})$. The distribution f_t is defined as $\boldsymbol{f}(\boldsymbol{x}_t, t, \boldsymbol{\theta})$, where $\boldsymbol{x}_t \sim q_t$, and the distribution g_t is defined as $\boldsymbol{g}(\boldsymbol{y}_t, t)$, where $\boldsymbol{y}_t \sim p_t$. The distribution q_t represents the noise distribution when generating samples.

Proof. The W-distance (Wasserstein-distance) is defined as follows:

$$\mathcal{W}_\rho[p, q] = \inf_{\gamma \in \prod[p,q]} \iint \gamma(\boldsymbol{x}, \boldsymbol{y}) \|\boldsymbol{x} - \boldsymbol{y}\|_\rho d\boldsymbol{x} d\boldsymbol{y},$$

where γ is any joint distribution of p and q. For convenience, we take the case of $\rho = 2$ and simply denote $\|\cdot\|$ as $\|\cdot\|_2$, and denote $\mathcal{W}[p, q]$ as $\mathcal{W}_2[p, q]$. Let $\{\boldsymbol{x}_{t_k}\}$ or $\{\boldsymbol{y}_{t_k}\}$ be the points on the same trajectory defined by the ODE in Eq. (1) on the ODE trajectory. For $\mathcal{W}[f_{t_k}, g_{t_k}]$, we have the following inequality:

$$\begin{aligned} &\mathcal{W}[f_{t_k}, g_{t_k}] \\ =&\inf_{\gamma^* \in \prod[f_{t_k}, g_{t_k}]} \iint \gamma^*(\hat{\boldsymbol{x}}_{t_k}, \hat{\boldsymbol{y}}_{t_k}) \|\hat{\boldsymbol{x}}_{t_k} - \hat{\boldsymbol{y}}_{t_k}\|_\rho d\hat{\boldsymbol{x}}_{t_k} d\hat{\boldsymbol{y}}_{t_k} \\ \overset{(i)}{\leq} &\iint \gamma(\hat{\boldsymbol{x}}_{t_k}, \hat{\boldsymbol{y}}_{t_k}) \|\hat{\boldsymbol{x}}_{t_k} - \hat{\boldsymbol{y}}_{t_k}\| d\hat{\boldsymbol{x}}_{t_k} d\hat{\boldsymbol{y}}_{t_k}, \gamma \in \prod[f_{t_k}, g_{t_k}] \\ =&\mathbb{E}_{\hat{\boldsymbol{x}}_{t_k}, \hat{\boldsymbol{y}}_{t_k} \sim \gamma \in \prod[f_{t_k}, g_{t_k}]}[\|\hat{\boldsymbol{x}}_{t_k} - \hat{\boldsymbol{y}}_{t_k}\|] \\ \overset{(ii)}{=} &\mathbb{E}_{\boldsymbol{x}_{t_k}, \boldsymbol{y}_{t_k} \sim \gamma \in \prod[q_{t_k}, p_{t_k}]}[\|\boldsymbol{f}(\boldsymbol{x}_{t_k}, t_k, \phi) - \boldsymbol{g}(\boldsymbol{y}_{t_k}, t_k)\|]. \end{aligned}$$

Here, (i) holds because γ is the joint distribution of any p_t and q_t. (ii) is obtained through the law of the unconscious statistician. Since the joint distribution $\gamma \in \prod[q_{t_k}, p_{t_k}]$ in the above formula is arbitrary, so we choose the distribution satisfying $\mathbb{E}_{\boldsymbol{x}_{t_k}, \boldsymbol{y}_{t_k} \sim \gamma^*}[\|\boldsymbol{y}_{t_k} - \boldsymbol{x}_{t_k}\|] = \mathcal{W}[q_{t_k}, p_{t_k}]$. We denote it as γ^*. The expectation $\mathbb{E}_{\boldsymbol{x}_{t_k}, \boldsymbol{y}_{t_k} \sim \gamma^*}[\|f(\boldsymbol{x}_{t_k}, t_k, \theta) - g(\boldsymbol{y}_{t_k}, t_k)\|]$ satisfies the following inequality:

$$\begin{aligned} &\mathbb{E}_{\boldsymbol{x}_{t_k}, \boldsymbol{y}_{t_k} \sim \gamma^*}[\|\boldsymbol{f}(\boldsymbol{x}_{t_k}, t_k, \boldsymbol{\theta}) - \boldsymbol{g}(\boldsymbol{y}_{t_k}, t_k)\|] \\ \leq &\mathbb{E}_{\boldsymbol{y}_{t_k} \sim p_{t_k}}[\|\boldsymbol{g}(\boldsymbol{y}_{t_k}, t_k) - \boldsymbol{f}(\boldsymbol{y}_{t_k}, t_k, \boldsymbol{\theta})\|] + L\mathcal{W}[q_{t_k}, p_{t_k}]. \end{aligned} \tag{6}$$

If the ODE solver is Euler ODE solver, we have:

$$\begin{aligned} &\mathbb{E}_{\boldsymbol{y}_{t_k} \sim p_{t_k}}[\|\boldsymbol{g}(\boldsymbol{y}_{t_k}, t_k) - \boldsymbol{f}(\boldsymbol{y}_{t_k}, t_k, \boldsymbol{\theta})\|] \\ \leq &\mathbb{E}_{\boldsymbol{y}_{t_{k-1}} \sim p_{t_{k-1}}}[\|\boldsymbol{g}(\boldsymbol{y}_{t_{k-1}}, t_{k-1}) - \boldsymbol{f}(\boldsymbol{y}_{t_{k-1}}, t_{k-1}, \boldsymbol{\theta})\|] \\ &+ L(t_{t_k} - t_{k-1})O(t_{t_k} - t_{k-1}) \\ &+ \mathbb{E}_{\boldsymbol{y}_{t_k} \sim p_{t_k}}[\|\boldsymbol{f}(\boldsymbol{y}_{t_{k-1}}^\phi, t_{k-1}, \boldsymbol{\theta}) - \boldsymbol{f}(\boldsymbol{y}_{t_k}, t_k, \boldsymbol{\theta})\|] \end{aligned} \tag{7}$$

The detailed proofs for the aforementioned inequalities can be found in Appendix B. We iterate multiple times until t_0. At this point, from Eq. (2), we have $\|g(y_{t_0}, t_0) - f(y_{t_0}, t_0, \theta)\| = 0$. So, we can obtain the inequality below:

$$\begin{aligned} &\mathbb{E}_{\boldsymbol{y}_{t_k} \sim p_{t_k}}[\|\boldsymbol{g}(\boldsymbol{y}_{t_k}, t_k) - \boldsymbol{f}(\boldsymbol{y}_{t_k}, t_k, \boldsymbol{\theta})\|] \\ \leq &\mathcal{L}_{CD}^k + \sum_{i=1}^k L(t_i - t_{i-1})O((t_i - t_{i-1})) \\ \overset{(i)}{=} &\mathcal{L}_{CT}^k + \sum_{i=1}^k t_k O((\Delta t)) + o(\Delta t). \end{aligned}$$

Here, (i) holds because $\Delta t = \max(t_k - t_{k-1})$, and the relationship between $\mathcal{L}_{CD}^k$ and $\mathcal{L}_{CT}^k$ in Eq. (3). Since consistency function $\boldsymbol{g}(\boldsymbol{x}_t, t) = \boldsymbol{x}_0$, it follows that $\mathcal{W}[f_{t_k}, g_{t_k}] = \mathcal{W}[f_{t_k}, p_0]$. Putting these together, the proof is complete. □

Analyzing Eq. (5), $\mathcal{W}[q_{t_k}, p_{t_k}]$ is the W-distance between the two sampling distributions, which is independent of the model. We set $q_t = p_t$ to eliminate $\mathcal{W}[q_{t_k}, p_{t_k}]$. The term $o(\Delta t)$ and $t_k O(\Delta t)$ originate from approximation errors, where $t_k O(\Delta t)$ increases with the increase of t_k. The remaining term is $\mathcal{L}_{CT}^k = \sum_{i=1}^k \mathbb{E}[d(f(\boldsymbol{x}_0 + t_i \boldsymbol{z}, t_i, \boldsymbol{\theta}), f(\boldsymbol{x}_0 + t_{i-1}\boldsymbol{z}, t_{i-1}, \boldsymbol{\theta}^-))]$. It can be seen that this term also accumulates errors. The quality of the model's generation depends not only on the current loss at t_k, $\mathbb{E}[d(f(\boldsymbol{x}_0 + t_k \boldsymbol{z}, t_k, \boldsymbol{\theta}), f(\boldsymbol{x}_0 + t_{k-1}\boldsymbol{z}, t_{k-1}, \boldsymbol{\theta}^-))]$, but also on the sum of all losses for values less than k. These two accumulated errors may be one of the reasons why consistency training requires as large a batch size and large model size as possible. During training, it is not only necessary to ensure a smaller loss at the current t_k, but also to use a larger batch size and larger model size to ensure a smaller loss at previous t values. Besides, reducing Δt can help to lower this upper bound. However, as described in the original text [41], reducing Δt in practical applications does not always lead to performance improvements.

3.3. Enhancing Consistency Training with Discriminator

Following the analysis in Sec. 3.2, it can be observed that the W-distance at time t_k depends not only on the loss at t_k, but also on the loss at previous times. This could be one of the reasons why consistency training requires as large a batch size and model size as possible. However, it can be noted that at each moment t_k, the ultimate goal is to reduce the distance between the generated distribution and the target distribution. In order to reduce the gap between two distributions, we propose not only using the W-distance, but also other distances, such as JS-divergence. Inspired by GANs, we suggest incorporating a discriminator into the training process.

It can be proven that when the generator training loss is given by

$$\mathcal{L}_G = \log(1 - D(\boldsymbol{f}(\boldsymbol{x} + t_{n+1}\boldsymbol{z}, t_{n+1}, \boldsymbol{\theta}_g), t_{n+1}, \boldsymbol{\theta}_d)), \tag{8}$$

and the discriminator training loss is given by

$$\begin{aligned}\mathcal{L}_D = &- \log(1 - D(\boldsymbol{f}(\boldsymbol{x}_g + t_{n+1}\boldsymbol{z}, t_{n+1}), \boldsymbol{\theta}_d) \\ &- \log(D(\boldsymbol{x}_r, t_{n+1}, \boldsymbol{\theta}_d)),\end{aligned} \tag{9}$$

minimizing the loss leads to $\min_{\boldsymbol{f}}(-2\log 2 + 2JSD\,(f_{t_k} \| p_0))$, which is equivalent to minimizing the JS-divergence. D is the discriminator. It can be observed that this loss does not depend on the previous t_k loss, and can directly optimize the distance between the current t_k distributions. Therefore, the required batch size and model size can be smaller compared to consistency training.

However, although the ultimate goals of the two distances are the same, e.g., when the JS-divergence is 0, the W-distance is also 0, at which point the gradient of the discriminator is also 0. However, at this point, the gradient of $\mathcal{L}_{CT}$ may not be 0 due to the aforementioned error. Moreover, when $\mathcal{L}_{CT}$ is relatively large, the optimization direction of $\mathcal{L}_{CT}$ may conflict with $\mathcal{L}_G$. Consider the extreme case where the output of f_{t_n} is completely random, it is clear that $\mathcal{L}_{CT}$ and $\mathcal{L}_G$ are in conflict, when training $\boldsymbol{f}$ at time t_{n+1}. On the other hand, when $\mathcal{L}_{CT}$ is relatively small, the model f is easier to fit at t_n than at t_{n+1}, thus generating better quality. Also, since x_t and $x_{t_{n+1}}$ are close enough, their discriminators are also close enough, thus jointly improving the generation quality. Therefore, we employ the coefficient λ to balance the proportion between $\mathcal{L}_{CT}$ and $\mathcal{L}_G$. Furthermore, as $\mathcal{L}_{CT}^k$ increases with k, the W-distance also increases. In order to improve the performance of consistency training, the weight of $\mathcal{L}_G$ should also increase. We utilize the formula Eq. (10) to give $\mathcal{L}_G$ more weight, where w is the weight at $n = N-1$, and w_{mid} is the weight at $n = (N-1)/2$.

$$\lambda_N(n) = w\left(\frac{n}{N-1}\right)^{\log_{\frac{1}{2}}(\frac{w_{mid}}{w})}. \tag{10}$$

Please note, even though the fitting targets of all f_{t_k} are q_0, we choose for the form $D(\boldsymbol{x}_t, t, \boldsymbol{\theta}_d)$ rather than $D(\boldsymbol{x}_t, \boldsymbol{\theta}_d)$ when constructing the discriminator. Although theoretically, the optimal distribution of the generator trained by these two discriminators is p_0, and for two similar samples, the discriminator in the form of $D(\boldsymbol{x}_t, \boldsymbol{\theta}_d)$ will generate similar gradients at different t, we find in our experiments Sec. 4.3.3 that this form of discriminator is not as effective as $D(\boldsymbol{x}_t, t, \boldsymbol{\theta}_d)$. The training algorithm is described in Algorithm 1.

Algorithm 1 Adversarial Consistency Training

```
1:  Input: dataset $\mathcal{D}$, initial consistency model parameter $\theta_g$, discriminator $\theta_d$, step schedule $N(\cdot)$, EMA decay rate schedule $\mu(\cdot)$, optimizer $\text{opt}(\cdot,\cdot)$, discriminator $D(\cdot,\cdot,\theta_d)$, adversarial rate schedule $\lambda(\cdot)$, gradient penalty weight $w_{gp}$, gradient penalty interval $I_{gp}$.
2:  $\boldsymbol{\theta}_g^- \leftarrow \boldsymbol{\theta}$ and $k \leftarrow 0$
3:  repeat
4:      Sample $\boldsymbol{x} \sim \mathcal{D}$, and $n \sim \mathcal{U}[\![1, N(k)]\!]$
5:      Sample $\boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I})$        ▷ Train Consistency Model
6:      $\mathcal{L}_{CT} \leftarrow$
            $d(\boldsymbol{f}(\boldsymbol{x} + t_{n+1}\boldsymbol{z}, t_{n+1}, \boldsymbol{\theta}_g), \boldsymbol{f}(\boldsymbol{x} + t_n\boldsymbol{z}, t_n, \boldsymbol{\theta}_g^-))$
7:      $\mathcal{L}_G \leftarrow$
            $\log(1 - D(\boldsymbol{f}(\boldsymbol{x} + t_{n+1}\boldsymbol{z}, t_{n+1}, \boldsymbol{\theta}_g), t_{n+1}, \boldsymbol{\theta}_d))$
8:      $\mathcal{L}_f \leftarrow (1 - \lambda_{N(k)}(n+1))\mathcal{L}_{CT} + \lambda_{N(k)}(n+1)\mathcal{L}_G$
9:      $\boldsymbol{\theta}_g \leftarrow \text{opt}(\boldsymbol{\theta}_g, \nabla_{\boldsymbol{\theta}_g}(\mathcal{L}_f))$
10:     $\boldsymbol{\theta}_g^- \leftarrow \text{stopgrad}(\mu(k)\boldsymbol{\theta}_g^- + (1 - \mu(k))\boldsymbol{\theta}_g)$

11:     Sample $\boldsymbol{x}_g \sim \mathcal{D}$, $\boldsymbol{x}_r \sim \mathcal{D}$, and $n \sim \mathcal{U}[\![1, N(k)]\!]$
12:     Sample $\boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I})$        ▷ Train Discriminator
13:     $\mathcal{L}_D \leftarrow -\log(D(\boldsymbol{x}_r, t_{n+1}, \boldsymbol{\theta}_d))$
            $-\log(1 - D(\boldsymbol{f}(\boldsymbol{x}_g + t_{n+1}\boldsymbol{z}, t_{n+1}, \boldsymbol{\theta}_d))$
14:     $\mathcal{L}_{gp} \leftarrow$
            $w_{gp}\|\nabla_{\boldsymbol{x}_r} D(\boldsymbol{x}_r, t_{n+1}, \boldsymbol{\theta}_d)\|^2[k \mod I_{gp} = 0]$
15:     $\mathcal{L}_d \leftarrow \lambda_{N(k)}(n+1)\mathcal{L}_D + \lambda_{N(k)}(n+1)\mathcal{L}_{gp}$
16:     $\boldsymbol{\theta}_d \leftarrow \text{opt}(\boldsymbol{\theta}_d, \nabla_{\boldsymbol{\theta}_d}(\mathcal{L}_d))$
17:     $k \leftarrow k + 1$
18: until convergence
```

3.4. Gradient Penalty Based Adaptive Data Augmentation

For smaller datasets, in the field of GANs, there are many data augmentation works to improve generation effects. Inspired by StyleGAN2-ADA[42], we also used adaptive differentiable data augmentation. However, unlike StyleGAN2-ADA, which adjusts the probability of data augmentation based on the accuracy of the discriminator over time, it is difficult to adjust the augmentation probability through the accuracy of a single discriminator in our model due to the varying training difficulties at different t. As described in Sec. 4.3.2, we find that the stability of the discriminator's gradient has a significant impact on training. This may be due to the interaction between $\mathcal{L}_{CT}$ and $\mathcal{L}_G$. We propose to adjust the probability of data augmentation based on the value

Table 1. Training steps and model parameter size are reported. BS stands for Batch Size. For ACT, Params represent parameters of the consistency model + discriminator.

Dataset	Method	BS	Steps	Params	Fid
CIFAR10	CT	512	800K	73.9M	8.7
	CT	256	800K	73.9M	10.4
	CT	128	800K	73.9M	14.4
	ACT-Aug	80	300K	27.5M+14.1M	**6.0**
ImageNet	CT	2048	800K	282M	13.0
	ACT	320	400K	107M+54M	**10.6**

Table 2. Sample quality of ACT on the ImageNet dataset with the resolution of 64×64. Our ACT significantly outperforms CT.

Method	NFE (↓)	FID (↓)	Prec. (↑)	Rec. (↑)
BigGAN-deep [3]	1	4.06	**0.79**	0.48
ADM [7]	250	**2.07**	0.74	0.63
EDM [20]	79	2.44	0.71	**0.67**
DDPM [18]	250	11.0	0.67	0.58
DDIM [38]	50	13.7	0.65	0.56
DDIM [38]	10	18.3	0.60	0.49
CT	1	13.0	**0.71**	0.47
ACT	1	**10.6**	0.67	**0.56**

of the gradient penalty over time. Given a differential data augmentation function $A(\boldsymbol{x}, p_{aug})$, where p_{aug} is the probability of applying the data augmentation, the augmented discriminator is defined by:

$$D_{aug}(\boldsymbol{x}_t, t, p_{aug}, \boldsymbol{\theta}_d) = D(A(\boldsymbol{x}_t, p_{aug}), t, \boldsymbol{\theta}_d).$$

The probability p_{aug} is updated by

$$p_{aug} \leftarrow \text{Clip}_{[0,1]}(p_{aug} + 2([\mathcal{L}_{gp}^{-} \geq \tau] - 0.5)p_r),$$

where $[\cdot]$ denotes the indicator function, which takes a value of 1 when the condition is true and 0 otherwise. $\text{Clip}_{[0,1]}(\cdot)$ represents the operation of clipping the value to the interval $[0, 1]$. p_r denotes the update rate at each iteration, and $\mathcal{L}_{gp}^{-}$ is the exponential moving average of $\mathcal{L}_{gp}$, defined as $\mathcal{L}_{gp}^{-} = \mu_p \mathcal{L}_{gp}^{-} + (1-\mu_p)\mathcal{L}_{gp}$. p_r and μ_p are constants within the range $[0, 1]$. This algorithm is described in Algorithm 2 shown in the Appendix. Our motivation for proposing the use of data augmentation is to mitigate the overfitting phenomenon in the discriminator. We conduct experiments on CIFAR10 to verify the method. However, the performance of data augmentation on large datasets, such as ImageNet 64×64, remains to be explored.

4. Experiments

In this section, we report some experimental settings and results on CIFAR-10 and ImageNet64 datasets.

4.1. Generation Performance

In this section, we report the performance of our model on the CIFAR10 and ImageNet 64×64 datasets. The results demonstrate a significant improvement of our method over the original approach. We exhibit the results on CIFAR10 in Tab. 3 and on ImageNet 64×64 in Tab. 2, respectively. The quality of generation on CIFAR10 improved from 8.7 to 6.0, while on ImageNet64, it improved from 13 to 10.6.

Furthermore, we demonstrate the performance of the consistency training on different batch sizes, as well as the sizes of the models used by the proposed method and consistency training, in Tab. 1. As can be discerned from the data in the table, the batch size has a significant impact on consistency training. When the batch size is set to 256, the FID score escalates to 10.4 from 8.7. Besides, with a batch size of 128, the FID rises to 14.4. On the CIFAR10 dataset, compared to consistency training, an FID score of 8.7 was achieved with a batch size of 512. The proposed method, however, achieved an FID score of 6.0 with a significantly smaller batch size of 80. On the ImageNet 64×64 dataset, the consistency training approach achieved an FID score of 13.0 with a batch size of 2048. In contrast, the proposed method achieved a lower FID score of 10.6, utilizing a notably smaller batch size of 320. Fig. 3 shows the generated samples from model training on ImageNet 64×64, and Appendix E provides explanations for all metrics.

4.2. Resource Consumption

We utilized the DDPM model architecture as our backbone. While DDPM's performance isn't as high as [7] and [40], it has fewer parameters and attention layers, enabling faster execution. Our model is significantly smaller than the 63.8M model used by consistency training on CIFAR10, with only 27.5M (41.6M with discriminator during training) parameters. On the ImageNet 64×64 dataset, our model, with only 107M parameters (161M with discriminator during training), is smaller than the 282M model used by consistency training. The smaller model and batch size reduce resource consumption. In our experiments on CIFAR10, we utilized 1 NVIDIA GeForce RTX 3090, as opposed to the 8 NVIDIA A100 GPUs used for consistency training. For the ImageNet 64×64 experiments, we employed 4 NVIDIA A100 GPUs, in contrast to the 64 A100 GPUs used for training in the consistency training setup.

4.3. Ablation Study

4.3.1 Impacts of λ_N

When $\lambda_N \equiv 0$, this reduces to consistency training. Conversely, when $\lambda_N \equiv 1$, it becomes Generative Adversarial Networks (GANs). According to the analysis in Sec. 3.2, as λ_N increases, adversarial consistency training gains the capacity to enhance model performance with smaller batch sizes, leveraging the discriminator. However, as discussed in Sec. 3.3, an overly large λ_N can lead to an excessive consistency training loss, thereby causing a conflict between $\mathcal{L}_{CT}$ and $\mathcal{L}_G$. Furthermore, it has been noted in the literature that for GANs, high-dimensional inputs may detrimentally affect model performance [30]. Therefore, as λ_N increases,

Table 3. Sample quality of ACT on the CIFAR10 dataset. We compare ACT with state-of-the-art GANs and (efficient) diffusion models. We show that ACT achieves the best FID and IS among all the one-step diffusion models.

Method	NFE (↓)	FID (↓)	IS (↑)
BigGAN [3]	1	14.7	9.22
AutoGAN [13]	1	12.4	8.40
ViTGAN [25]	1	6.66	9.30
TransGAN [19]	1	9.26	9.05
StyleGAN2-ADA [42]	1	2.92	**9.83**
StyleGAN2-XL [37]	1	**1.85**	-
Score SDE [40]	2000	2.20	**9.89**
DDPM [18]	1000	3.17	9.46
EDM [20]	36	**2.04**	9.84
DDIM [38]	50	4.67	-
DDIM [38]	20	6.84	-
DDIM [38]	10	8.23	-
1-Rectified Flow [27]	1	378	1.13
Glow [21]	1	48.9	3.92
Residual FLow [4]	1	46.4	-
DenseFlow [15]	1	34.9	-
DC-VAE [31]	1	17.9	8.20
CT [41]	1	8.70	8.49
ACT	1	6.4	8.93
ACT-Aug	1	**6.0**	**9.15**

the model performance exhibits a pattern of initial improvement followed by a decline. Firstly, we demonstrate the phenomenon of mode collapse when $\lambda_N \approx 1$ on CIFAR10. As illustrated in Fig. 1, the phenomenon of mode collapse is observed. It can be noted that, apart from the initial t_k where the residual structure from Eq. (2) results in outputs with substantial input components, preventing mode collapse, the other t_k values all exhibit mode collapse.

For a score-based model as defined in Sec. 3.1.1, the learned sampling process is the reverse of the diffusion process $p_t(\boldsymbol{x}_0|\boldsymbol{x}_t)$. However, the distribution $q_t(\boldsymbol{x}_0|\boldsymbol{x}_t)$ learned via Eqs. (8) and (9) does not consider the forward process of the diffusion. We conduct further experiments where the form of the discriminator is changed to $D(\boldsymbol{x}_0, \boldsymbol{x}_t, t, \boldsymbol{\theta}_d)$, and it can be proven Appendix H that the distribution learned by the generator is $p_t(\boldsymbol{x}_0|\boldsymbol{x}_t)$. However, we also observe the phenomenon of mode collapse in our experiments. Fig. 4 illustrates the training collapse on ImageNet 64×64 when $\lambda_N \equiv 0.3$. It can be observed that at around 150k training steps, the $\mathcal{L}_{CT}$ becomes unstable and completely collapses around 170k. We have included the training curves for the proper λ_N in the Appendix G. It can be observed that at this point, $\mathcal{L}_{CT}$ and several other training losses remain stable. Essentially, a smaller w_{mid} and a larger w are preferable choices.

4.3.2 Connection between gradient penalty and training stability

In Sec. 3.3, we analyzed the relationship between $\mathcal{L}_{CT}$ and $\mathcal{L}_G$, highlighting the importance of gradient stability. In this section, we conduct experiments to validate our previous analysis and demonstrate the rationality of the ACT-Aug method proposed in Sec. 3.4.

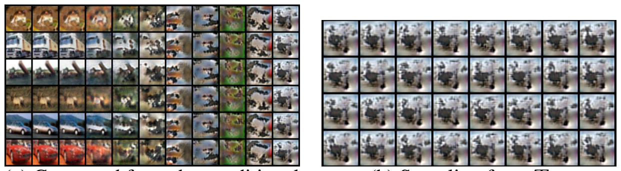
(a) Generated from the conditional trajectory $\{\boldsymbol{x}_0 + t_k \boldsymbol{z}\}$. (b) Sampling from $T\boldsymbol{z}$.

Figure 1. Failed generations. Mode collapse when $\lambda_N \approx 1$. Experiments are conducted on the CIFAR10 dataset.

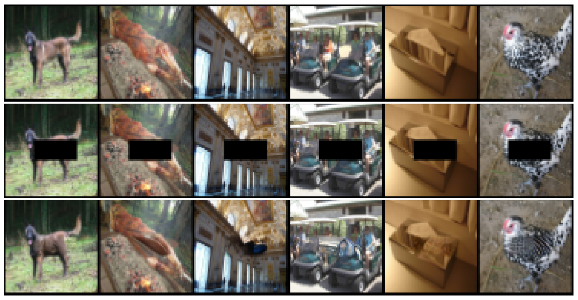
Figure 2. The results of zero-shot inpainting. **First Row:** original images; **Second Row:** masked images; **Bottom Row:** inpainted images.

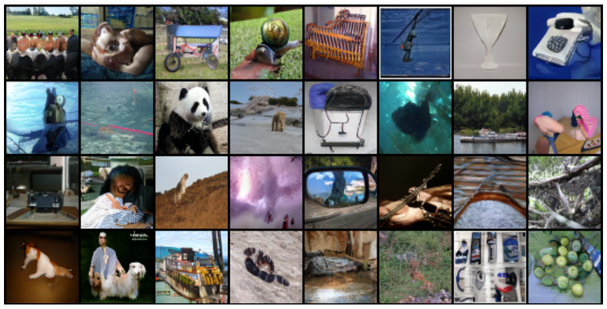
Figure 3. Generated samples (ACT Trained on ImageNet 64×64).

Fig. 4 illustrates the relationship among the values of the gradient penalty ($\mathcal{L}_{gp}$), consistency training loss ($\mathcal{L}_{CT}$), and FID. It can be observed that almost every instance of instability in $\mathcal{L}_{CT}$ is accompanied by a relatively large $\mathcal{L}_{gp}$. Fig. 5 illustrates the relationship among these three on the CIFAR10 dataset. It can be seen that in the mid-stage of training, $\mathcal{L}_{gp}$ begins to slowly increase, a process that is accompanied by a gradual increase in $\mathcal{L}_{CT}$ and FID. Therefore, we believe that gradient stability is crucial for adversarial consistency training. Based on this, we propose ACT-Aug (Sec. 3.4) on small datasets, using $\mathcal{L}_{gp}$ as an indicator to adjust the probability of data augmentation, thereby stabilizing $\mathcal{L}_{gp}$ around a certain value.

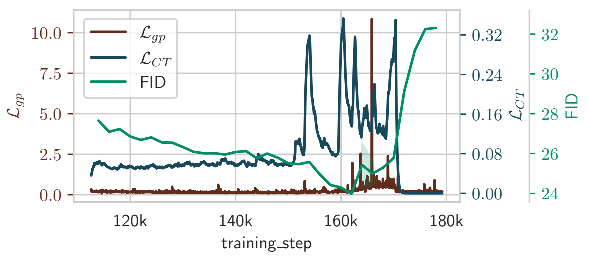

Figure 4. $\mathcal{L}_{gp}$, $\mathcal{L}_{CT}$, and FID of ACT on ImageNet 64x64 ($\lambda_N \equiv 0.3$, an overly large λ_N leads to training collapse. Additionally, drastic changes in $\mathcal{L}_{gp}$ closely follow changes in $\mathcal{L}_{CT}$).

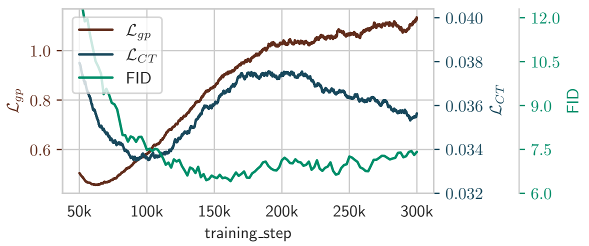

Figure 5. $\mathcal{L}_{gp}$, $\mathcal{L}_{CT}$, and FID of ACT on CIFAR10 ($\lambda_N \equiv 0.3$, this is an appropriate λ_N. In the later stages of training, without data augmentation, $\mathcal{L}_{CT}$, $\mathcal{L}_{gp}$, and FID all show relatively large increases).

4.3.3 Discriminator

Activation Function We have chosen to compare LeakyReLU and SiLU. Generally, GANs employ LeakyReLU as the activation function for the discriminator. This function is typically considered to provide better gradients for the generator. On the other hand, SiLU is the activation function chosen for DDPM, and it is generally regarded as a stronger activation function compared to LeakyReLU. Tab. 4 displays the FID scores of different activation functions on CIFAR10 at 50k and 150k training steps. Contrary to previous findings, we discovery that utilizing the SiLU function for the discriminator led to faster convergence rates and improved final performance. A possible reason is that $\mathcal{L}_{CT}$ provides an additional gradient direction, which mitigates the overfitting of the discriminator.

Different Backbone Tab. 4 also displays the FID scores of different architecture on CIFAR10 at 50k and 150k training steps. In our investigation, we have evaluated the discriminators of StyleGAN2, ProjectedGAN and the downsampling part of DDPM (simply denoted as DDPM) as described in Appendix A. Due to the significant role of residual structures in designing GANs' discriminators, we incorporate residual connections between different downsampling blocks in DDPM, denoted as DDPM-res. It can be observed that DDPM performs the best. Although DDPM-res exhibits a faster convergence rate during the early stages of training, its performance in the later stages is not as satisfactory as that of DDPM. Furthermore, we find that DDPM demonstrates superior training stability compared to DDPM-res. We also experiment with whether or not to feed t into the discriminator, denoted as t-emb. We find that feeding t yields better results. This might be due to the fact that the optimal value of the discriminator varies with different t_k, hence the necessity of t-emb for better fitting.

Table 4. Ablation study of the discriminator.

Discriminator	Activation	t-emb	Fid (50k)	Fid (150k)
DDPM-res	LeakyReLU	False	18.7	10.6
DDPM-res	LeakyReLU	True	11.5	7.4
DDPM-res	SiLU	True	**9.9**	7.0
DDPM	SiLU	True	12.5	**6.5**
StyleGAN2	LeakyReLU	True	16.7	9.5
ProjectedGAN	LeakyReLU	True	19.4	16.6

4.4. Zero-shot Image Inpainting

An important capability of consistency models is zero-shot image inpainting. This depends on the properties of the diffusion process and $\mathcal{L}_{CT}$. Given that we introduce a discriminator during the training process, does this impact the properties of consistency models? We demonstrate the results of inpainting in Fig. 2. We employ the algorithm consistent with [41]. It can be seen that ACT still retains the capabilities of consistency models.

In Appendix D, we further display the sampling results from the conditional trajectory $\{\boldsymbol{x}_0 + t_k\boldsymbol{z}\}, \boldsymbol{x}_0 \sim p_0, \boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I})$ on ImageNet 64×64. k ranges from 0 to N, with 10 equidistant points. It can be observed that the sampling results of t_k and t_{k-1} exhibit significant similarity, which further substantiates that ACT does not disrupt the properties of $\mathcal{L}_{CT}$ and consistency models.

5. Conclusion

We have proposed Adversarial Consistency Training (ACT), an improvement over consistency training. Through analyzing the consistency training loss, which is proven to be the upper bound of the W-distance between the sampling and target distributions, we have introduced a method that directly employs Jensen-Shannon Divergence to minimize the distance between the generated and target distributions. This approach enables superior generation quality with less than $1/6$ of the original batch size and approximately $1/2$ of the original model parameters and training steps, thereby having smaller resource consumption. Our method retains the beneficial capabilities of consistency models, such as inpainting. Additionally, we propose to use gradient penalty-based adaptive data augmentation to improve the performance on small datasets. The effectiveness of our approach has been validated on CIFAR10 and ImageNet 64×64 datasets, highlighting its potential for broader application in the field of image generation.

However, the interaction between $\mathcal{L}_{CT}$ and $\mathcal{L}_G$ can be further explored so as to further improve our method. In addition to using JS-Divergence, other distances can also be used to reduce the distance between the generated and target distributions. In the future, we will focus on these two aspects to further boost the performance of our method.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017. 2

[2] Shane Barratt and Rishi Sharma. A note on the inception score. *arXiv: Machine Learning*, abs/1801.01973, 2018. 2

[3] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019. 1, 6, 7

[4] Ricky T. Q. Chen, Jens Behrmann, David Duvenaud, and J&ouml;rn-Henrik Jacobsen. Residual flows for invertible generative modeling. In *Conference on Neural Information Processing Systems*, pages 9913–9923, 2019. 1, 7

[5] christian szegedy, vincent vanhoucke, sergey ioffe, jonathon shlens, and zbigniew wojna. Rethinking the inception architecture for computer vision. *Proceedings - IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, abs/1512.00567(1):2818–2826, 2016. 2

[6] Giannis Daras, Yuval Dagan, Alexandros G Dimakis, and Constantinos Daskalakis. Consistent diffusion models: Mitigating sampling drift by learning to be consistent. *arXiv preprint arXiv:2302.09057*, 2023. 2

[7] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021. 6

[8] Tim Dockhorn, Arash Vahdat, and Karsten Kreis. Score-based generative modeling with critically-damped langevin diffusion. In *International Conference on Learning Representations*, 2022. 2

[9] Chris Donahue, Julian McAuley, and Miller Puckette. Adversarial audio synthesis. *arXiv preprint arXiv:1802.04208*, 2018. 2

[10] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. In *International Conference on Learning Representations*, 2017. 2

[11] Jinhao Duan, Fei Kong, Shiqi Wang, Xiaoshuang Shi, and Kaidi Xu. Are diffusion models vulnerable to membership inference attacks? *International Conference on Machine Learning*, 2023. 1

[12] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martín Arjovsky, Olivier Mastropietro, and Aaron C. Courville. Adversarially learned inference. In *International Conference on Learning Representations*, 2017. 2

[13] Xinyu Gong, Shiyu Chang, Yifan Jiang, and Zhangyang Wang. Autogan: Neural architecture search for generative adversarial networks. In *IEEE International Conference on Computer Vision*, pages 3223–3233, 2019. 7

[14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. 1, 2

[15] Matej Grcić, Ivan Grubišić, and Siniša Šegvić. Densely connected normalizing flows. In *Conference on Neural Information Processing Systems*, pages 23968–23982, 2021. 7

[16] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017. 2

[17] 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 *Conference on Neural Information Processing Systems*, 2017. 2

[18] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, pages 6840–6851, 2020. 1, 6, 7

[19] Yifan Jiang, Shiyu Chang, and Zhangyang Wang. Transgan: Two pure transformers can make one strong gan, and that can scale up. *Advances in Neural Information Processing Systems*, 34:14745–14758, 2021. 7

[20] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Conference on Neural Information Processing Systems*, 2022. 6, 7

[21] Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Conference on Neural Information Processing Systems*, 2018. 7

[22] Fei Kong, Jinhao Duan, RuiPeng Ma, Hengtao Shen, Xiaofeng Zhu, Xiaoshuang Shi, and Kaidi Xu. An efficient membership inference attack for the diffusion model by proximal initialization. *arXiv preprint arXiv:2305.18355*, 2023. 1

[23] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations*, 2021. 1

[24] Tuomas Kynk&auml;&auml;nniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *Advances in neural information processing systems*, 32:3929–3938, 2019. 2

[25] Kwonjoon Lee, Huiwen Chang, Lu Jiang, Han Zhang, Zhuowen Tu, and Ce Liu. Vitgan: Training gans with vision transformers. In *International Conference on Learning Representations*, 2022. 7

[26] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *ICLR 2023*, 2022. 1

[27] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *ICLR*, 2023. 7

[28] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11461–11471, 2022. 1

[29] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative ad-

versarial networks. In *International Conference on Learning Representations*, 2018. 2

[30] Manisha Padala, Debojit Das, and Sujit Gujar. Effect of input noise dimension in gans. In *Neural Information Processing: 28th International Conference, ICONIP 2021, Sanur, Bali, Indonesia, December 8–12, 2021, Proceedings, Part III 28*, pages 558–569. Springer, 2021. 6

[31] Gaurav Parmar, Dacheng Li, Kwonjoon Lee, and Zhuowen Tu. Dual contradistinctive generative autoencoder. *Proceedings - IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 823–832, 2021. 7

[32] Vadim Popov, Ivan Vovk, Vladimir Gogoryan, Tasnima Sadekova, and Mikhail Kudinov. Grad-tts: A diffusion probabilistic model for text-to-speech. In *International Conference on Machine Learning*, pages 8599–8608, 2021. 1

[33] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *user-5f6bfffd92c7f9be21bbcc99*, 2022. 2

[34] 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 10674–10685, 2022. 2

[35] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022. 2

[36] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, 2016. 2

[37] Axel Sauer, Katja Schwarz, and Andreas Geiger. Stylegan-xl: Scaling stylegan to large diverse datasets. In *International Conference on Computer Graphics and Interactive Techniques*, pages 1–10, 2022. 7

[38] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. 2, 6, 7

[39] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, pages 11895–11907, 2019. 1

[40] 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. 1, 2, 3, 6, 7

[41] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *Computing Research Repository*, abs/2303.01469, 2023. 1, 2, 3, 4, 7, 8

[42] Karras Tero, Aittala Miika, Hellsten Janne, Laine Samuli, Lehtinen Jaakko, and Aila Timo. Training generative adversarial networks with limited data. In *Conference on Neural Information Processing Systems*, pages 12104–12114, 2020. 2, 5, 7

[43] Karras Tero, Laine Samuli, Aittala Miika, Hellsten Janne, Lehtinen Jaakko, and Aila Timo. Analyzing and improving the image quality of stylegan. In *Computer Vision and Pattern Recognition*, pages 8107–8116, 2020. 2

[44] Hoang Thanh-Tung, Truyen Tran, and Svetha Venkatesh. Improving generalization and stability of generative adversarial networks. In *International Conference on Learning Representations*, 2019. 2

[45] Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. `https://github.com/huggingface/diffusers`, 2022. 1

[46] Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion gans. In *International Conference on Learning Representations*, 2022. 2

[47] Chenxi Yuan and Mohsen Moghaddam. Attribute-aware generative design with generative adversarial networks. *IEEE Access*, 8:190710–190721, 2020. 2

[48] Chenxi Yuan, Jinhao Duan, Nicholas J Tustison, Kaidi Xu, Rebecca A Hubbard, and Kristin A Linn. Remind: Recovery of missing neuroimaging using diffusion models with application to alzheimer's disease. *medRxiv*, pages 2023–08, 2023. 1

[49] Chenxi Yuan, Tucker Marion, and Mohsen Moghaddam. Ddegan: Integrating a data-driven design evaluator into generative adversarial networks for desirable and diverse concept generation. *Journal of Mechanical Design*, 145(4):041407, 2023. 2

[50] Han Zhang, Zizhao Zhang, Augustus Odena, and Honglak Lee. Consistency regularization for generative adversarial networks. *ICLR*, 2020. 1

[51] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient gan training. In *Conference on Neural Information Processing Systems*, pages 7559–7570, 2020. 2

[52] Hongkai Zheng, Weili Nie, Arash Vahdat, Kamyar Azizzadenesheli, and Anima Anandkumar. Fast sampling of diffusion models via operator learning. *ICML 2023*, 2023. 1

ACT: Adversarial Consistency Models

Supplementary Material

A. Architecture

For the consistency model architecture, we employ a structure similar to that of DDPM [18], with the exception of altering the corresponding embeddings to continuous time. We utilize the Python library diffusers [45]. In terms of the discriminator, we employ the downsampling structure in the DDPM, preserving it up to the mid-block. Subsequently, a linear layer is added to map it to $\mathbb{R}$. Additionally, the layers-per-block parameter is set to 150% of that in the consistency model, with all other parameters remaining the same. The parameters passed to the UNet2DModel are listed in Tab. 5. B=128. In the context of block type, 'D' represents DownBlock2D, 'A' stands for either AttnDownBlock2D or AttnUpBlock2D, and 'U' means UpBlock2D.

	CIFAR10	ImageNet 64×64
layers_per_block	2	2
block_out_channels	(1B,1B,2B,2B)	(1B,2B,2B,4B,4B)
down_block_types	DADD	DDADD
up_block_types	UUAU	UUAUU
attention_head_dim	8	16

Table 5. The parameters passed to the UNet2DModel. For those not listed, the default settings from the diffusers library are used.

B. Details of the Proof for Theorem 3.1

Details for Eq. (6):

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{x}_{t_k},\boldsymbol{y}_{t_k}\sim\gamma^*}[\|\boldsymbol{f}(\boldsymbol{x}_{t_k},t_k,\boldsymbol{\theta})-\boldsymbol{g}(\boldsymbol{y}_{t_k},t_k)\|]\\
=&\mathbb{E}_{\boldsymbol{x}_{t_k},\boldsymbol{y}_{t_k}\sim\gamma^*}[\|\boldsymbol{g}(\boldsymbol{y}_{t_k},t_k)-\boldsymbol{f}(\boldsymbol{y}_{t_k},t_k,\boldsymbol{\theta})\\
&\qquad+\boldsymbol{f}(\boldsymbol{y}_{t_k},t_k,\boldsymbol{\theta})-\boldsymbol{f}(\boldsymbol{x}_{t_k},t_k,\boldsymbol{\theta})\|]\\
\leq&\mathbb{E}_{\boldsymbol{x}_{t_k},\boldsymbol{y}_{t_k}\sim\gamma^*}[\|\boldsymbol{g}(\boldsymbol{y}_{t_k},t_k)-\boldsymbol{f}(\boldsymbol{y}_{t_k},t_k,\boldsymbol{\theta})\|\\
&\qquad+\|\boldsymbol{f}(\boldsymbol{y}_{t_k},t_k,\boldsymbol{\theta})-\boldsymbol{f}(\boldsymbol{x}_{t_k},t_k,\boldsymbol{\theta})\|]\\
\overset{(i)}{\leq}&\mathbb{E}_{\boldsymbol{x}_{t_k},\boldsymbol{y}_{t_k}\sim\gamma^*}[\|\boldsymbol{g}(\boldsymbol{y}_{t_k},t_k)-\boldsymbol{f}(\boldsymbol{y}_{t_k},t_k,\boldsymbol{\theta})\|\\
&\qquad+L\|\boldsymbol{y}_{t_k}-\boldsymbol{x}_{t_k}\|]\\
=&\mathbb{E}_{\boldsymbol{x}_{t_k},\boldsymbol{y}_{t_k}\sim\gamma^*}[\|\boldsymbol{g}(\boldsymbol{y}_{t_k},t_k)-\boldsymbol{f}(\boldsymbol{y}_{t_k},t_k,\boldsymbol{\theta})\|]\\
&\qquad+L\mathbb{E}_{\boldsymbol{x}_{t_k},\boldsymbol{y}_{t_k}\sim\gamma^*}[\|\boldsymbol{y}_{t_k}-\boldsymbol{x}_{t_k}\|]\\
=&\mathbb{E}_{\boldsymbol{y}_{t_k}\sim p_{t_k}}[\|\boldsymbol{g}(\boldsymbol{y}_{t_k},t_k)-\boldsymbol{f}(\boldsymbol{y}_{t_k},t_k,\boldsymbol{\theta})\|]+L\mathcal{W}[q_{t_k},p_{t_k}].
\end{aligned}$$

Here, (i) holds because $\boldsymbol{f}$ satisfies the Lipschitz condition.

Details for Eq. (7):

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{y}_{t_k}\sim p_{t_k}}[\|\boldsymbol{g}(\boldsymbol{y}_{t_k},t_k)-\boldsymbol{f}(\boldsymbol{y}_{t_k},t_k,\boldsymbol{\theta})\|]\\
\overset{(i)}{=}&\mathbb{E}_{\boldsymbol{y}_{t_k}\sim p_{t_k}}[\|\boldsymbol{g}(\boldsymbol{y}_{t_{k-1}},t_{k-1})-\boldsymbol{f}(\boldsymbol{y}_{t_{k-1}},t_{k-1},\boldsymbol{\theta})\\
&\quad+\boldsymbol{f}(\boldsymbol{y}_{t_{k-1}},t_{k-1},\boldsymbol{\theta})-\boldsymbol{f}(\boldsymbol{y}^{\phi}_{t_{k-1}},t_{k-1},\boldsymbol{\theta})\\
&\qquad+\boldsymbol{f}(\boldsymbol{y}^{\phi}_{t_{k-1}},t_{k-1},\boldsymbol{\theta})-\boldsymbol{f}(\boldsymbol{y}_{t_k},t_k,\boldsymbol{\theta})\|]\\
\leq&\mathbb{E}_{\boldsymbol{y}_{t_k}\sim p_{t_k}}[\|\boldsymbol{g}(\boldsymbol{y}_{t_{k-1}},t_{k-1})-\boldsymbol{f}(\boldsymbol{y}_{t_{k-1}},t_{k-1},\boldsymbol{\theta})\|]\\
&\qquad+\mathbb{E}_{\boldsymbol{y}_{t_k}\sim p_{t_k}}[\|\boldsymbol{f}(\boldsymbol{y}_{t_{k-1}},t_{k-1},\boldsymbol{\theta})-\boldsymbol{f}(\boldsymbol{y}^{\phi}_{t_{k-1}},t_{k-1},\boldsymbol{\theta})\|]\\
&\qquad+\mathbb{E}_{\boldsymbol{y}_{t_k}\sim p_{t_k}}[\|\boldsymbol{f}(\boldsymbol{y}^{\phi}_{t_{k-1}},t_{k-1},\boldsymbol{\theta})-\boldsymbol{f}(\boldsymbol{y}_{t_k},t_k,\boldsymbol{\theta})\|]\\
\overset{(ii)}{\leq}&\mathbb{E}_{\boldsymbol{y}_{t_k}\sim p_{t_k}}[\|\boldsymbol{g}(\boldsymbol{y}_{t_{k-1}},t_{k-1})-\boldsymbol{f}(\boldsymbol{y}_{t_{k-1}},t_{k-1},\boldsymbol{\theta})\|]\\
&\qquad+L\|\boldsymbol{y}_{t_{k-1}}-\boldsymbol{y}^{\phi}_{t_{k-1}}\|\\
&\qquad+\mathbb{E}_{\boldsymbol{y}_{t_k}\sim p_{t_k}}[\|\boldsymbol{f}(\boldsymbol{y}^{\phi}_{t_{k-1}},t_{k-1},\boldsymbol{\theta})-\boldsymbol{f}(\boldsymbol{y}_{t_k},t_k,\boldsymbol{\theta})\|]\\
\overset{(iii)}{=}&\mathbb{E}_{\boldsymbol{y}_{t_{k-1}}\sim p_{t_{k-1}}}[\|\boldsymbol{g}(\boldsymbol{y}_{t_{k-1}},t_{k-1})-\boldsymbol{f}(\boldsymbol{y}_{t_{k-1}},t_{k-1},\boldsymbol{\theta})\|]\\
&\qquad+L(t_{t_k}-t_{k-1})O(t_{t_k}-t_{k-1})\\
&\qquad+\mathbb{E}_{\boldsymbol{y}_{t_k}\sim p_{t_k}}[\|\boldsymbol{f}(\boldsymbol{y}^{\phi}_{t_{k-1}},t_{k-1},\boldsymbol{\theta})-\boldsymbol{f}(\boldsymbol{y}_{t_k},t_k,\boldsymbol{\theta})\|]
\end{aligned}$$

Here, (i) holds because $\boldsymbol{g}$ is a consistency function, with $\boldsymbol{g}(\boldsymbol{y}_{t_k},t_k)=\boldsymbol{g}(\boldsymbol{y}_{t_{k-1}},t_{k-1})$. (ii) holds because $\boldsymbol{f}$ satisfies the Lipschitz condition. (iii) holds because Φ is an Euler solver, hence $\|\boldsymbol{y}_{t_{k-1}}-\boldsymbol{y}^{\phi}_{t_{k-1}}\|$ does not exceed the truncation error $O((t_n-t_{n-1})^2)$.

C. ACT-Aug

In this section, we will provide the details of ACT-Aug. The differences from ACT are highlighted in red. The algorithm is listed in Algorithm 2.

D. Generation Visualization on Conditional Trajectory

In this section, we demonstrate samples generated from the conditional trajectory $\{\boldsymbol{x}_0+t_k\boldsymbol{z}\}$ on ImageNet 64×64, further illustrating that our method preserves the properties of consistency training. Fig. 6 shows the conditional trajectory $\{\boldsymbol{x}_0+t_k\boldsymbol{z}\}$, while Fig. 7 displays the samples generated from the conditional trajectory $\{\boldsymbol{x}_0+t_k\boldsymbol{z}\}$. It can be observed that there is a high degree of similarity between adjacent t values, further validating that our method retains the properties of $\mathcal{L}_{CT}$.

Algorithm 2 Adversarial Consistency Training with Augmentation

```
1: Input: dataset $\mathcal{D}$, initial consistency model parameter $\theta_g$, discriminator $\theta_d$, step schedule $N(\cdot)$, EMA decay rate schedule $\mu(\cdot)$, optimizer $\text{opt}(\cdot,\cdot)$, discriminator with augmentation $D_{aug}(\cdot,\cdot,\cdot,\theta_d)$, adversarial rate schedule $\lambda(\cdot)$, gradient penalty weight $w_{gp}$, gradient penalty interval $I_{gp}$, gradient penalty threshold $\tau$, augmentation probability update rate $p_r$
2: $\boldsymbol{\theta}_g^- \leftarrow \boldsymbol{\theta}$, $k \leftarrow 0$, $p_{aug} \leftarrow 0$ and $\mathcal{L}_{gp}^- = \tau$
3: repeat
4:     Sample $\boldsymbol{x} \sim \mathcal{D}$, and $n \sim \mathcal{U}[\![1, N(k)]\!]$
5:     Sample $\boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I})$        ▷ Train Consistency Model
6:     $\mathcal{L}_{CT} \leftarrow$
           $d(\boldsymbol{f}(\boldsymbol{x} + t_{n+1}\boldsymbol{z}, t_{n+1}, \boldsymbol{\theta}_g), \boldsymbol{f}(\boldsymbol{x} + t_n\boldsymbol{z}, t_n, \boldsymbol{\theta}_g^-))$
7:     $\mathcal{L}_G \leftarrow \log(1-$
           $D_{aug}(\boldsymbol{f}(\boldsymbol{x} + t_{n+1}\boldsymbol{z}, t_{n+1}, p_{aug}, \boldsymbol{\theta}_g), t_{n+1}, \boldsymbol{\theta}_d))$
8:     $\mathcal{L}_f \leftarrow (1 - \lambda_{N(k)}(n+1))\mathcal{L}_{CT} + \lambda_{N(k)}(n+1)\mathcal{L}_G$
9:     $\boldsymbol{\theta}_g \leftarrow \text{opt}(\boldsymbol{\theta}_g, \nabla_{\boldsymbol{\theta}_g}(\mathcal{L}_f))$
10:    $\boldsymbol{\theta}_g^- \leftarrow \text{stopgrad}(\mu(k)\boldsymbol{\theta}_g^- + (1 - \mu(k))\boldsymbol{\theta}_g)$

11:    Sample $\boldsymbol{x}_g \sim \mathcal{D}$, $\boldsymbol{x}_r \sim \mathcal{D}$, and $n \sim \mathcal{U}[\![1, N(k)]\!]$
12:    Sample $\boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I})$        ▷ Train Discriminator
13:    $\mathcal{L}_D \leftarrow -\log(D_{aug}(\boldsymbol{x}_r, t_{n+1}, p_{aug}, \boldsymbol{\theta}_d))$
           $-\log(1 - D_{aug}(\boldsymbol{f}(\boldsymbol{x}_g + t_{n+1}\boldsymbol{z}, t_{n+1}, p_{aug}, \boldsymbol{\theta}_d))$
14:    $\mathcal{L}_{gp} \leftarrow w_{gp}[k \mod I_{gp} = 0] *$
           $\|\nabla_{\boldsymbol{x}_r} D_{aug}(\boldsymbol{x}_r, t_{n+1}, p_{aug}, \boldsymbol{\theta}_d)\|^2$
15:    $\mathcal{L}_d \leftarrow \lambda_{N(k)}(n+1)\mathcal{L}_D + \lambda_{N(k)}(n+1)\mathcal{L}_{gp}$
16:    $\boldsymbol{\theta}_d \leftarrow \text{opt}(\boldsymbol{\theta}_d, \nabla_{\boldsymbol{\theta}_d}(\mathcal{L}_d))$
17:    if $k \mod I_{gp} = 0$ then
18:        $p_{aug} \leftarrow$
               $\text{Clip}_{[0,1]}(p_{aug} + 2([\mathcal{L}_{gp}^- >= \tau] - 0.5)p_r)$
19:        $\mathcal{L}_{gp}^- = \mu_p\mathcal{L}_{gp}^- + (1 - \mu_p)\mathcal{L}_{gp}$
20:    end if
21:    $k \leftarrow k + 1$
22: until convergence
```

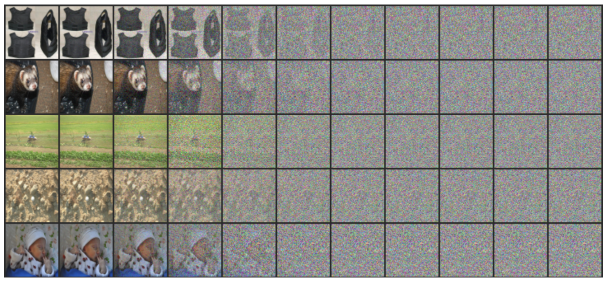
Figure 6. The conditional trajectory $\{\boldsymbol{x}_0 + t_k\boldsymbol{z}\}$ (ImageNet 64×64).

E. Metrics

The metrics utilized are IS, FID, Improved Precision and Improved Recall. The Inception Score (IS), introduced in [36], assesses a model's ability to generate convincing images of distinct ImageNet classes and capture the overall class distribution. However, it has a limitation in that it doesn't incentivize capturing the full distribution or the diversity within classes, leading to models with high IS even if they only memorize a small portion of the dataset, as noted in [2]. To address the need for a metric that better reflects diversity, the Fréchet Inception Distance (FID) was introduced in [17]. This metric is argued to align more closely with human judgment than IS, and it quantifies the similarity between two image distributions in the latent space of Inception-V3 as detailed in [5]. Additionally, [24] developed Improved Precision and Recall metrics that evaluate the fidelity of generated samples by determining the proportion that aligns with the data manifold (precision) and the diversity by the proportion of real samples that are represented in the generated sample manifold (recall).

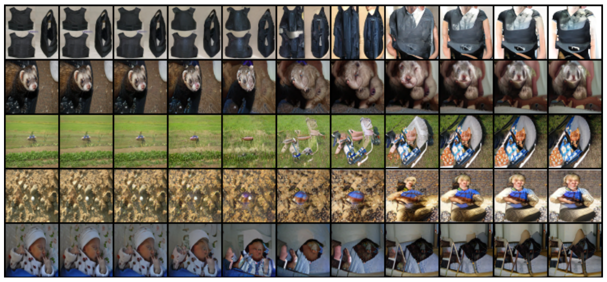
Figure 7. Generated from the conditional trajectory $\{\boldsymbol{x}_0 + t_k\boldsymbol{z}\}$ (ImageNet 64×64).

F. Experiment settings

In this section, we report the configuration of various hyperparameters within our experimental framework. Tab. 6 provides a summary of the experimental setup. Unless otherwise specified, the learning rate for both the consistency model and the discriminator is identical. The experiments conducted during the ablation study (Sec. 4.3), maintained consistency with the settings outlined in this table, with the exception of the parameters specifically varied for the ablation study. Additionally, when employing the ProjectedGAN as the discriminator, the learning rate of discriminator is set to 0.002, with w and w_{mid} values at 0.1.

G. Examples of proper λ_N

In this section, we present the stability of $\mathcal{L}_{CT}$, $\mathcal{L}_{gp}$, and the FID score of the appropriate selection of λ_N. As depicted in Fig. 8, it is observed that all three metrics exhibit stability during training. Specifically for $\mathcal{L}_{gp}$, there is an initial decreasing trend followed by an increase; however, the variation remains within a range of 0.1 until the end of training.

Hyperparameter	CIFAR10	ImageNet 64×64
Discriminator	DDPM	DDPM
Learning rate	1e-4	5e-5
Batch size	80	320
μ_0	0.9	0.95
s_0	2	2
s_1	150	200
w	0.3	0.2
w_{mid}	0.3	0.6
I_{gp}	16	16
τ	0.55	-
p_r	0.05	-
μ_p	0.93	-
Training iterations	300k	400k
Mixed-Precision	No	Yes
Number of GPUs	1×RTX 3090	4×A100

Table 6. Summary of the experimental setup.

Fig. 9 illustrates the stability of $\mathcal{L}_{gp}$, $\mathcal{L}_{CT}$, and the FID score for ACT-Aug under the appropriate selection of λ_N. It is observed that all three metrics exhibit stability. Furthermore, when compared with ACT on CIFAR10 as shown in Fig. 5, $\mathcal{L}_{gp}$ is stabilized around the set $\tau = 0.55$, and both $\mathcal{L}_{CT}$ and the FID score continue to show a decreasing trend. This validates the effectiveness of the augmentation.

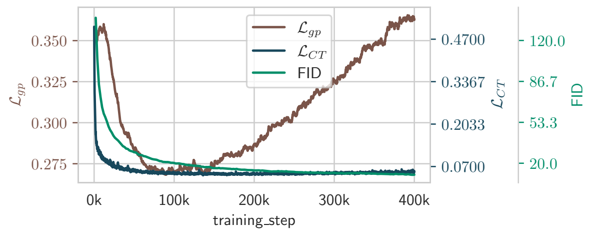

Figure 8. $\mathcal{L}_{gp}$, $\mathcal{L}_{CT}$, and FID of ACT on ImageNet 64x64 ($w_{mid=0.2}, w = 0.6$, a suitable parameter set. Under these parameters, all three metrics demonstrate stability).

H. Conditional Discriminator

Theorem H.1. *Given a generator $G(\boldsymbol{z}, \boldsymbol{x}_t, t)$ and a discriminator $D(\boldsymbol{x}_0, \boldsymbol{x}_t, t)$. The distribution of optimal solution of $G(\cdot, \boldsymbol{x}_t, t)$ for the problem Eq. (11) is $p_g(\cdot|\boldsymbol{x}_t) = p(\cdot|\boldsymbol{x}_t)$, where $p_g(\cdot|\boldsymbol{x}_t)$ is the sample distribution of $G(\boldsymbol{z}, \boldsymbol{x}_t, t)$, $z \sim p_{\boldsymbol{z}}(\boldsymbol{z}|\boldsymbol{x}_t)$. $p_{\boldsymbol{z}}(\cdot|\boldsymbol{x}_t)$ is a normal distribution. $\boldsymbol{x}_t \sim p_t$, and $\boldsymbol{x}_0 \sim p_0$. p_t is the marginal distribution of a diffusion process.*

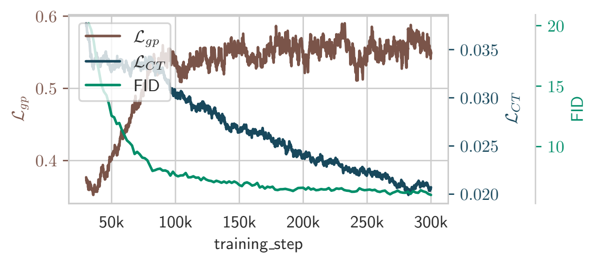

Figure 9. $\mathcal{L}_{gp}$, $\mathcal{L}_{CT}$, and FID of ACT-Aug on CIFAR10 ($\lambda_N \equiv 0.3$, a suitable parameter set. Under these parameters, all three metrics demonstrate stability).

$$\min_G \max_D V(G, D) = \mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{x}_t \sim p(\boldsymbol{x}_0, \boldsymbol{x}_t)}[\log D(\boldsymbol{x}_0, \boldsymbol{x}_t)] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z}|\boldsymbol{x}_t), \boldsymbol{x}_t \sim p_t}[\log(1 - D(G(\boldsymbol{z}, \boldsymbol{x}_t, t), \boldsymbol{x}_t))] \tag{11}$$

Proof. By expressing Eq. (11) in integral form, we have the following equation:

$$\begin{aligned}
&\iint_{\boldsymbol{x}_0, \boldsymbol{x}_t} p(\boldsymbol{x}_0, \boldsymbol{x}_t) \log(D(\boldsymbol{x}_0, \boldsymbol{x}_t)) d\boldsymbol{x}_0 d\boldsymbol{x}_t \\
&+ \iint_{\boldsymbol{z}, \boldsymbol{x}_t} p_{\boldsymbol{z}}(\boldsymbol{z}, \boldsymbol{x}_t) \log(1 - D(G(\boldsymbol{z}, \boldsymbol{x}_t), \boldsymbol{x}_t)) d\boldsymbol{z} d\boldsymbol{x}_t \\
= &\int_{\boldsymbol{x}_t} p_t(\boldsymbol{x}_t) \left(\int_{\boldsymbol{x}_0} p(\boldsymbol{x}_0|\boldsymbol{x}_t) \log(D(\boldsymbol{x}_0, \boldsymbol{x}_t)) d\boldsymbol{x}_0 \right. \\
&\left. + \int_{\boldsymbol{z}} p_{\boldsymbol{z}}(\boldsymbol{z}|\boldsymbol{x}_t) \log(1 - D(G(\boldsymbol{z}, \boldsymbol{x}_t), \boldsymbol{x}_t)) d\boldsymbol{z} \right) d\boldsymbol{x}_t \\
= &\mathbb{E}_{\boldsymbol{x}_t \sim p_t} \left[\int_{\boldsymbol{x}_0} p(\boldsymbol{x}_0|\boldsymbol{x}_t) \log(D(\boldsymbol{x}_0, \boldsymbol{x}_t)) \right. \\
&\left. + p_g(\boldsymbol{x}_0|\boldsymbol{x}_t) \log(1 - D(\boldsymbol{x}_0, \boldsymbol{x}_t)) d\boldsymbol{x}_0 \right]
\end{aligned}$$

The optimal D is:

$$D_G^* = \frac{p(\boldsymbol{x}_0|\boldsymbol{x}_t)}{p(\boldsymbol{x}_0|\boldsymbol{x}_t) + p_g(\boldsymbol{x}_0|\boldsymbol{x}_t)}$$

Substituting D^* into V, we obtain the following equation:

$$\begin{aligned}
&\max_D V(G, D) \\
= &\mathbb{E}_{\boldsymbol{x}_t \sim p_t} \left[\mathbb{E}_{\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0|\boldsymbol{x}_t)} \left[\log \frac{p(\boldsymbol{x}_0|\boldsymbol{x}_t)}{p(\boldsymbol{x}_0|\boldsymbol{x}_t) + p_g(\boldsymbol{x}_0|\boldsymbol{x}_t)} \right] \right. \\
&\left. + \mathbb{E}_{\boldsymbol{x}_0 \sim p_g(\boldsymbol{x}_0|\boldsymbol{x}_t)} \log \left[\frac{p_g(\boldsymbol{x}_0|\boldsymbol{x}_t)}{p(\boldsymbol{x}_0|\boldsymbol{x}_t) + p_g(\boldsymbol{x}_0|\boldsymbol{x}_t)} \right] \right] \\
= &\mathbb{E}_{\boldsymbol{x}_t \sim p_t} [-\log 4 + 2 JSD(p_t(\cdot|\boldsymbol{x}_t) || p_g(\cdot|\boldsymbol{x}_t))]
\end{aligned}$$

In the aforementioned equation, *JSD* represents the Jensen-Shannon divergence. The equation holds true only when $p_g(\cdot|\boldsymbol{x}_t) = p(\cdot|\boldsymbol{x}_t)$. This concludes the proof. □