ONE-STEP IMAGE-FUNCTION GENERATION VIA CONSISTENCY TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Consistency models aim to deliver a U-Net generator to map noise to images directly and enable swift inference with minimal steps, even trained in isolation with consistency training mode. However, the U-Net generator requires heavy feature extraction layers for multi-level resolutions and learning convolution kernels with specific receptive fields, resulting in the challenge that consistency models suffer from heavy training resources and fail to generate images with any user-specific resolutions. In this paper, we first validate that training the original consistency model with a small batch size via consistency training mode is pretty unstable, which motivates us to investigate efficient and flexible consistency models. To this end, we propose to use a novel Transformer-based generator to generate *continuous image functions*, which can then be differentially rendered as images with arbitrary resolutions. We adopt implicit neural representations (INRs) to form such continuous functions, which help to decouple the resolution of generated images and the total amount of the parameters generated from the neural network. Extensive experiments on one-step image generation demonstrate that our method greatly improves the performance of consistency models with low training resources and also provides an efficient any-resolution image sampling process.

1 INTRODUCTION

Diffusion models Sohl-Dickstein et al. (2015); Song & Ermon (2019; 2020) have achieved remarkable efficacy in synthesizing various signals, including audio Kong et al. (2020); Chen et al. (2020), image Dhariwal & Nichol (2021); Ramesh et al. (2022) and video Harvey et al. (2022); Ho et al. (2022). However, diffusion models rely on an iterative sampling process, leading to slow generation. Consistency model Song et al. (2023) is an emerging family of diffusion models Ho et al. (2020); Song et al. (2020a) that directly map noise to data by maintaining point consistency on ODE trajectory. This unique characteristic enables consistency models to support rapid one-step generation for high-quality samples by design. Consistency models Song et al. (2023) can be trained through two modes: consistency distillation from a pre-trained diffusion model or consistency training in isolation. Different from consistency distillation, consistency training does not rely on the pre-trained diffusion model but utilizes an unbiased estimator to approximate the ground-truth score function Song et al. (2023). Consequently, consistency training emerges as a more flexible and convenient method for training consistency models and shows greater potential in generation tasks.

However, consistency models face challenges due to their substantial training resource requirement and inflexible image generation with fixed resolution. Consistency models rely on a U-Net generator Ronneberger et al. (2015) to map noise to images, while the U-Net involves extensive convolution to extract features and is proved to be less scalable than the Transformer-based generator in diffusion models Peebles & Xie (2023). Our investigation reveals that training consistency models based on a U-Net generator under the consistency training model requires a large batch size. Directly reducing the training batch size to accommodate limited training resources significantly diminishes the generation performance, leading to the generation of non-realistic images, as illustrated in Figure 1 (a). Besides, as depicted in Figure 1 (b), to generate a specific-resolution image, the U-Net needs to denoise a noisy image with the same resolution, causing consistency models can only generate images with fixed resolution once trained on a dataset with specific-resolution images. Therefore, a more efficient and flexible generator is desired for consistency models.

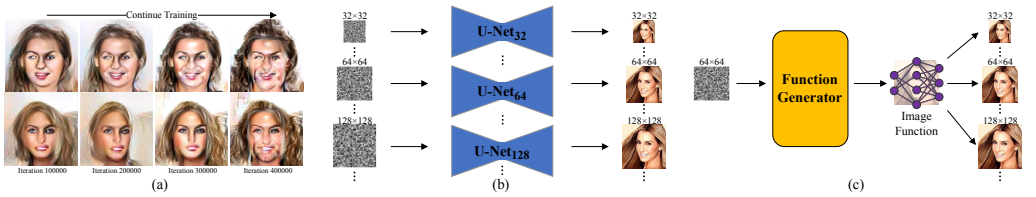


Figure 1: (a) When we train the consistency models on the CelebA dataset with consistency training and a batch size of 4, the generation performance gradually diminishes. (b) The U-Net generates images with the same resolution as the noisy images. Therefore we need to train infinite separate generators at all resolutions if we want to sample images with arbitrary resolutions. (c) Our image function generator treats the images as continuous functions parameterized as the MLPs. With a single generator, it generates a fixed amount of MLP parameters as an image function, which can then be rendered as images with arbitrary resolutions.

In this paper, we propose a novel and efficient generator for consistency models that stabilizes the consistency training process under low training resources and can generate images with any user-specific resolution in the inference phase. Specifically, instead of generating discrete grid representations of images, we treat images as continuous functions and introduce a novel Transformer-based generator Dosovitskiy et al. (2020) that predicts the continuous functions of images as intermediates. These intermediates can then be differentially rendered into images with arbitrary resolution, as depicted in Figure 1 (c). To represent the continuous functions of images, we leverage implicit neural representations Xie et al. (2022) (INRs) that employ Multi-layer Perceptrons (MLPs) to map the coordinates $x \in \mathbb{R}^2$ to corresponding RGB values $y \in \mathbb{R}^3$. Compared to the U-Net generator, our Transformer-based generator exhibits a more stable consistency training process of consistency models with limited training resources. Moreover, the use of differential rendering enables the decoupling of the resolution of the generated images and the total amount of parameters generated from the neural network, leading to efficient sampling of images at arbitrary resolutions.

Additionally, we empirically confirm that consistency training from scratch to generate functions of high-resolution images is challenging and converges slowly. To enhance training efficiency, we carefully design a novel architecture for the function generator, which consists of a feature extraction module to denoise the noisy images, a function prediction module that predicts functions for clean images, and a non-learnable render module to obtain clean images. We then introduce an image reconstruction pre-training task to pre-train the function prediction module, which compels the generator to convert a given clean image to its corresponding INR function, mitigating the optimization challenges in consistency training when predicting the function of the clean image based solely on a given noisy image.

We summarize the contributions of this work as follows:

- Toward efficient and flexible one-step generation for any-resolution images with consistency models, we propose a novel Transformer-based function generator, which first generates image functions with a single inference step, and then renders the image functions to produce images with arbitrary resolutions.
- We carefully design a novel end-to-end architecture for the function generator that simplifies the generation of functions for clean images from given noisy images. This architecture involves a feature extraction module, a function prediction module, and a render module.
- To enhance the consistency training efficiency, we introduce an image reconstruction pre-training task to fortify the function prediction module, enabling the function prediction module to predict functions of images giving clean images. This approach allows consistency training process to concentrate on the image-denoising task, resulting in more realistic image generation and faster convergence of the training process.
- Extensive experiments on one-step image generation under low training resources demonstrate that our methods can generate significantly more realistic images with much less training and inference resources than the original consistency models.

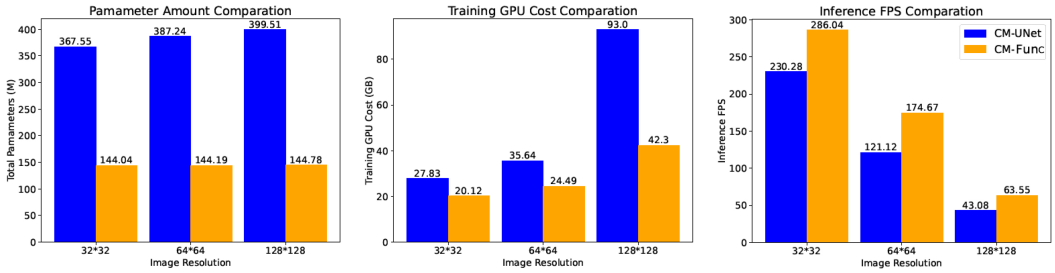


Figure 2: The comparison of the consistency training efficiency with the U-Net generator and our function generator based on image functions, including the parameter amount (left, lower is better), the training GPU costs (middle, lower is better), and inference frame per second (FPS, right, higher is better). We can observe that the increment of the parameter amount of our generator is nearly negligible, while the training GPU cost and inference FPS of our function generator are more efficient and have better scalability than the U-Net generator.

2 Related Work

Diffusion Models. Diffusion models Sohl-Dickstein et al. (2015); Song et al. (2020a) are emerging topics in the computer vision community. They train a generator to denoise the noise-corrupted data to estimate the score of data distribution and iteratively denoise the data point sampled from the noise distribution to generate new samples. Lots of work are done to accelerate the inference speed of diffusion models, such as faster ODE solver Song et al. (2020a); Lu et al. (2022a;b), predictor-corrector methods Song et al. (2020b) distillation methods Salimans & Ho (2022); Meng et al. (2023) Yin et al. (2024) and rectification Liu et al. (2022; 2023c).

Consistency Models. Consistency models Song et al. (2023) is a new type of diffusion model for few-step sampling while maintaining good generation quality. They deliver consistency mapping to map any point in ODE trajectory to its origin, enabling one-step generation. Unlike GAN-based generation models Goodfellow et al. (2014), consistency models do not rely on adversarial optimization and thus avoid the associated training difficulty. Consistency models can be trained either by consistency distillation mode from a pre-trained diffusion model or by consistency training mode with an unbiased estimator to approximate the ground-truth score function. Luo *et al.* Luo et al. (2023) apply consistency distillation to train consistency models in latent space. In this work, we focus on consistency training because it does not rely on an additional pre-trained diffusion model and is more flexible and convenient for training.

Efficient Diffusion. Apart from classical U-Net Ronneberger et al. (2015), several works Bao et al. (2023a;b); Peebles & Xie (2023) successfully adopt Vision Transformer Dosovitskiy et al. (2020) as the generator in diffusion models. They Bao et al. (2023a); Peebles & Xie (2023) show that the Transformer generator enjoys better scalability and higher performance in generating high-resolution images in latent diffusion models Rombach et al. (2022) than the U-Net generator. However, these works only explore the simple case of replacing the U-Net generator in the diffusion models with a ViT generator. On the contrary, our work is the first work to deliver a ViT generator to generate INR functions in the novel consistency models that support one-step generation.

Flexible Image Generation. Some works have targeted the problem of flexible image generation with any resolution, either based on modifying the diffusion trajectory and model or based on a patch-by-patch assembly manner. Haji-Ali et al. (2023); Zhang et al. (2023) focus on the flexible sampling of U-Net-based diffusion models and enable iteratively generating images with specific resolutions by decoupling the generation trajectory or dynamically adjusting the feature map size. However, these methods rely on operating on the iterative sampling process, which makes them unsuitable for the single-step sampling process with consistency models. The patch-by-patch assembly manner works Chai et al. (2022); Lin et al. (2022) deliver a patch-by-patch strategy to generate patches and assemble the patches into a larger image with particular resolution. However, the output of their generator is still of a fixed resolution, therefore they require multiple inference processes for a larger image and cannot generate images with lower resolution.

Implicit Neural Representations. By mapping a coordinate to its corresponding quantity with a neural network (e.g. MLP), INRs have shown great potential in representing complex continuous functions for a lot of natural signals, such as time-serial signals Fons et al. (2022); Szatkowski et al., images Sitzmann et al. (2020b); Skorokhodov et al. (2021); Liu et al. (2023a) and 3D scenes Park et al. (2019); Sitzmann et al. (2020a); Liu et al. (2023b). A lot of works on how to generate INRs fast for unseen signals have been employed, including meta-learning Sitzmann et al. (2020a); Tancik et al. (2021); Liu et al. (2023a); Finn et al. (2017) and hyper-network Chen & Wang (2022); Kim et al. (2023); Zhang et al. (2024).

Diffusion Models Based on Implicit Neural Representations. Different from diffusion based on explicit field Zhuang et al. (2023), when adopting diffusion models to high-resolution images or complex 3D signals, existing novel research Dupont et al. (2022); Karnewar et al. (2023); Erkoç et al. (2023); Chen et al. (2024) firstly convert the signals to their corresponding INR functions and then train a diffusion model on these INR functions, enabling generating complex signals efficiently. However, the two-stage training process is inflexible and the error in the first representation stage would greatly affect the performance of the second diffusion stage. On the contrary, our work proposes an end-to-end training framework to generate INRs of signals and requires rendering for only one time when generating one-step inference results.

3 INR-BASED CONSISTENCY TRAINING

3.1 PRELIMINARIES

Consistency models Song et al. (2023) is a new family of generative models that enables a few-step generation. The core idea of CM is the PF-ODE Song et al. (2020b). Denote the data distribution by $p_{\text{data}}(\mathbf{x})$, and the perturbed distribution is presented as $p_\sigma(\mathbf{x}) = \int p_{\text{data}}(\mathbf{y}) \mathcal{N}\left(\mathbf{x} \mid \mathbf{y}, \sigma^2 \boldsymbol{I}\right) \mathrm{d}\mathbf{y}$ if we add Gaussian noise $\mathcal{N}(0, \sigma^2)$ with noise level σ to the data. Then, the PF-ODE presented in Karras *et al.* Karras et al. (2022) is formulated as:

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\sigma} = -\sigma \nabla \log p_\sigma(\mathbf{x}) \approx -\sigma \boldsymbol{s}_\phi(\mathbf{x}, \sigma), \tag{1}$$

where $\boldsymbol{s}_\phi(\mathbf{x}, \sigma) \approx \nabla \log p_\sigma(\mathbf{x})$ is the *score function* Song et al. (2020b) of $p_\sigma(\mathbf{x})$. Here, as in Song et al. (2023); Karras et al. (2022), σ is defined as $\sigma \in [\sigma_{\min}, \sigma_{\max}]$, where $\sigma_{\min}$ is a small positive number to ensure $p_{\sigma_{\min}}(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x})$ and $\sigma_{\max}$ is a large positive number such that $p_{\sigma_{\max}}(\mathbf{x}) \approx \mathcal{N}\left(\mathbf{0}, \sigma_{\max}^2 \mathbf{I}\right)$.

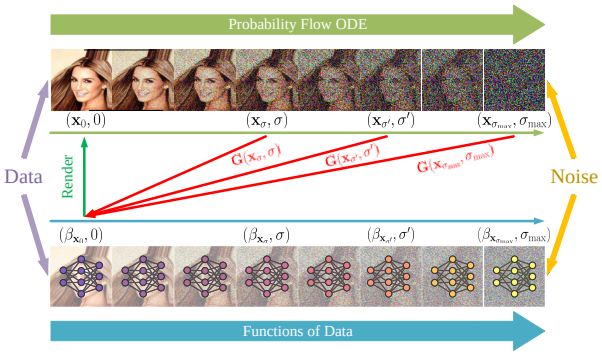


Figure 3: Given a probability-flow ODE that smoothly converts data to noise, we learn a denoising neural network G to map any point (e.g., $\mathbf{x}_\sigma$, $\mathbf{x}_{\sigma'}$, and $\mathbf{x}_{\sigma_{\max}}$) on the ODE trajectory to the continuous function of the origin (e.g., $\beta_{\mathbf{x}_0}$) for generative modeling, which can then be differentially rendered as the original data with arbitrary resolutions.

Solving the PF-ODE from noise level σ to σ_{min} in Eq. 1 indeed establishes a bijective mapping from a noisy data sample $\mathbf{x}_\sigma \sim p_\sigma(\mathbf{x})$ to the real data sample $\mathbf{x}_{\sigma_{\min}} \sim p_{\sigma_{\min}}(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x})$. This mapping $\boldsymbol{f}^* : (\mathbf{x}_\sigma, \sigma) \mapsto \mathbf{x}_{\sigma_{\min}}$ is defined as a *consistency function* in Song *et al.* Song et al. (2023). By the definition, the consistency function satisfies the *boundary condition* $\boldsymbol{f}^*(\mathbf{x}, \sigma_{\min}) = \mathbf{x}$. To approximate the consistency function with boundary condition, a consistency model $\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}, \sigma)$ is parameterized as:

$$\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}, \sigma) = c_{\text{skip}}(\sigma)\mathbf{x} + c_{\text{out}}(\sigma)\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}, \sigma), \tag{2}$$

where $\boldsymbol{F}_{\boldsymbol{\theta}}(\mathrm{x}, \sigma)$ is a free-form denoising neural network with parameter $\boldsymbol{\theta}$, while $c_{\text{skip}}(\sigma)$ and $c_{\text{out}}(\sigma)$ are differential functions so that $c_{\text{skip}}(\sigma_{\min}) = 1$ and $c_{\text{out}}(\sigma_{\min}) = 0$.

The consistency function has the property of *self-consistency*: the outputs are consistent for arbitrary pairs of $(\mathbf{x}_\sigma, \sigma)$ that belong to the same PF-ODE trajectory, which can be formulated as:

$$\boldsymbol{f}(\mathbf{x}_\sigma, \sigma) = \boldsymbol{f}(\mathbf{x}_{\sigma'}, \sigma'), \forall \sigma, \sigma' \in [\sigma_{\min}, \sigma_{\max}]. \tag{3}$$

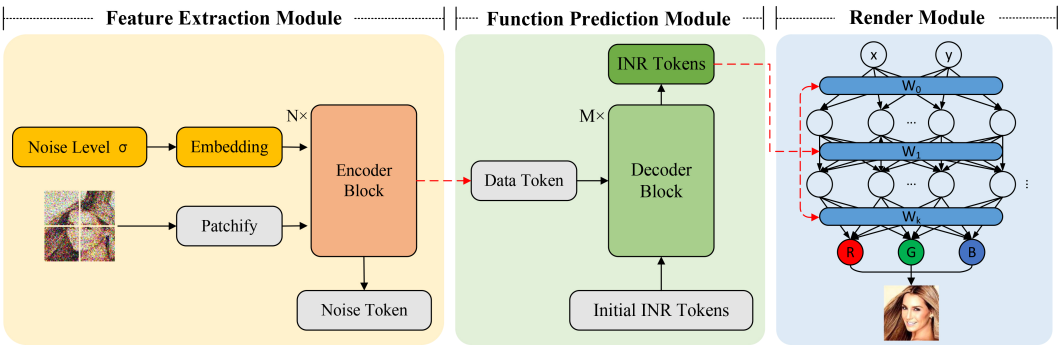


Figure 4: The architecture for our image-function generator. It contains three modules: the feature extraction module that contains several encoder blocks to extract features on the input noisy images and gives data tokens as a condition to guide the generation of the image function; the function prediction module that contains several decoder blocks to predict the INR parameters for image functions with given data token from the previous module; the non-learnable render module that multiples the INR parameters with coordinates and finally render as images.

Therefore, consistency models can be trained by enforcing the self-consistency property with a consistency loss between the results denoised from the i^{th} noise level and the $(i+1)^{th}$ noise level:

$$\mathcal{L} = \mathbb{E}\left[d\left(\boldsymbol{f}_{\boldsymbol{\theta}}\left(\mathbf{x}_{\sigma_{i+1}}, \sigma_{i+1}\right), \boldsymbol{f}_{\boldsymbol{\theta}^-}\left(\hat{\mathbf{x}}^{\phi}_{\sigma_i}, \sigma_i\right)\right)\right], \tag{4}$$

where $d(\cdot,\cdot)$ is a metric function such as ℓ_2 metric or learned perceptual image patch similarity (LPIPS) metric Zhang et al. (2018) while $\boldsymbol{\theta}^- \leftarrow \mu\boldsymbol{\theta}^- + (1-\mu)\boldsymbol{\theta}$ is the target model parameter updated with the exponential moving average (EMA) of the parameter $\boldsymbol{\theta}$ and EMA decay rate μ. $\hat{\mathbf{x}}^{\phi}_{\sigma_i}$ is derived from $\mathbf{x}_{\sigma_{i+1}}$ by solving the PF-ODE in the reverse direction for a single step:

$$\hat{\mathbf{x}}_{\sigma_i} = \mathbf{x}_{\sigma_{i+1}} - \left(\sigma_i - \sigma_{i+1}\right)\sigma_{i+1}\nabla_{\mathbf{x}} \log p_{\sigma_{i+1}}(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_{\sigma_{i+1}}}.$$

To estimate the unknown score function $\nabla_{\mathbf{x}} \log p_{\sigma_{i+1}}(\mathbf{x})$, Song *et al.* Song et al. (2023) propose *consistency training* that employs an approximation $\hat{\mathbf{x}}_{\sigma_i} = \mathbf{x} + \sigma_i \mathbf{z}$ with the same $\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$ and $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ to calculate $\mathbf{x}_{\sigma_{i+1}} = \mathbf{x} + \sigma_{i+1}\mathbf{z}$. Therefore, the consistency training objective $\mathcal{L}_{\text{CT}}$ can be defined as:

$$\mathcal{L}_{\text{CT}} = \mathbb{E}\left[d\left(\boldsymbol{f}_{\boldsymbol{\theta}}\left(\mathbf{x} + \sigma_{i+1}\mathbf{z}, \sigma_{i+1}\right), \boldsymbol{f}_{\boldsymbol{\theta}^-}\left(\mathbf{x} + \sigma_i \mathbf{z}, \sigma_i\right)\right)\right]. \tag{5}$$

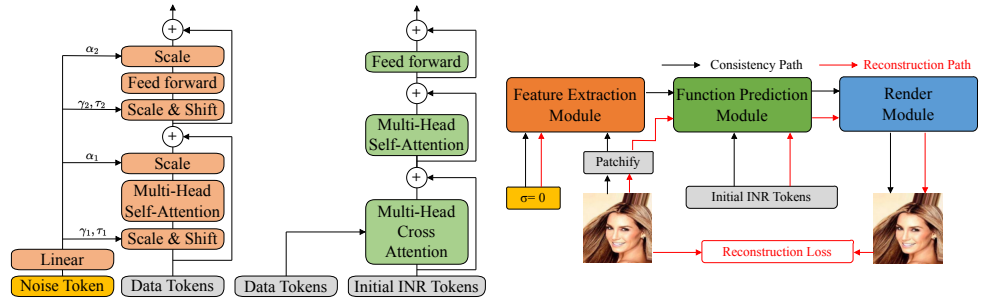
3.2 Generate Image Functions and Sample Any-resolution Images

Our modification focuses on the free-form denoising neural network $\boldsymbol{F}_{\boldsymbol{\theta}} : (\mathbf{x}, \sigma) \rightarrow \mathbf{x}_d$, which is typically a U-Net model Song & Ermon (2019); Ronneberger et al. (2015) that gets noisy image $\mathbf{x} \in \mathcal{R}^{C\times H\times W}$ and noise level $\sigma \in \mathcal{R}$ as input. As shown in Figure 1 (b), the U-Net model applies heavy feature extraction at the different resolution of the image and finally outputs a denoised image $\mathbf{x}_d \in \mathcal{R}^{C\times H\times W}$. We find that the U-Net can only generate the output image with exactly the same resolution as the input image, which is not flexible enough to scale to high-resolution images with limited training resources.

Therefore, we seek a more efficient and more noise-robust method to generate high-resolution images Dupont et al. (2022); Rahaman et al. (2019); Skorokhodov et al. (2021). We show our pipeline in Figure 3. As in INR methods Sitzmann et al. (2020b); Liu et al. (2023a); Chen & Wang (2022), rather than discrete grid representation, we consider images as continuous functions, which can be parameterized as neural networks, e.g. MLPs Dupont et al. (2022). Specifically, we consider an image as a collection of paired coordinates and RGB values $\{(\mathbf{c}, \mathbf{y})\}_{H\times W}$ and fit an MLP $\boldsymbol{M}_{\beta}$ with learnable parameter β to map the coordinates $\mathbf{c} \in \mathcal{R}^2$ to its corresponding RGB values $\mathbf{y} \in \mathcal{R}^3$:

$$\boldsymbol{M}_{\beta}(\mathbf{c}) = \mathbf{y}. \tag{6}$$

To obtain smooth super-resolution performance and eliminate artifacts, we follow Zhang et al. (2024) to apply variational coordinates to sample the coordinates $\mathbf{c}$.



(a) Encoder Block Architecture (b) Decoder Block Architecture (c) Pre-Training Data Flow

Figure 5: (a) & (b) The detailed implementation of encoder and decoder block (Norm is not shown). (c) Pre-training data flow. We set the noise level to 0, skip the feature extraction module, and optimize the Function Prediction Module to predict the image function for clean images with reconstruction loss. The black path is the data flow when consistency training the whole model while the red path is the pre-training of the function prediction module with the image reconstruction task.

Note that each image can be considered as an image function $\boldsymbol{M}_\beta$ and holds its own parameter $\beta = \{\mathbf{W}^t, \mathbf{B}^t\}_{t=0}^{t=T-1}$ for an MLP with totally T layers. Generating a denoised image is equivalent to generating the INR parameter β of the image function corresponding to the denoised image:

$$\boldsymbol{G}_\theta(\mathbf{x}, \sigma) = \beta, \tag{7}$$

where $\boldsymbol{G}_\theta$ is our proposed function generator and its architecture is introduced in Section 3.3.

Once the parameter β is obtained, the denoised image $\mathbf{x}_d$ can be reconstructed at arbitrary resolution by querying the RGB values with any specific coordinates $\{\mathbf{c}\}_{H\times W}$. As a result, we parameterize the free-form neural network $\boldsymbol{F}_\theta$ with a generator that generates image function as intermediate:

$$\boldsymbol{F}_\theta(\mathbf{x}, \sigma, \{\mathbf{c}\}) = \boldsymbol{M}_{\boldsymbol{G}_\theta(\mathbf{x},\sigma)}(\{\mathbf{c}\}) = \mathbf{x}_d. \tag{8}$$

Note that each layer of MLP can be formulated as:

$$\mathbf{c}^{t+1} = \texttt{act}(\mathbf{W}^t\mathbf{c}^t + \mathbf{B}^t), \tag{9}$$

where $\texttt{act}$ is the activation function. The whole forward process of $\boldsymbol{M}_\beta(\mathbf{c})$ is entirely differential if the activation function $\texttt{act}$ is differential, which enables the end-to-end backward process after calculating consistency training loss, as shown in Eq. 5.

Compared with the U-Net generator that generates images with the same resolution as the input images, our function generator only generates the INR parameters β of the image function that has a fixed size and does not scale with the input image resolution. When scaling to the larger resolution image, our pipeline only needs to query the image function with finer-grained coordinates $\{\mathbf{c}\}$, which greatly decreases the parameters amount, training time, and memory cost of the whole pipeline and provides a flexible any-resolution image sampling process.

3.3 INR Generator Design

In this part, we introduce the detailed architecture of our function generator. As presented in Figure 4, it contains three major modules: a feature extraction module, a function prediction module, and a render module.

Feature Extraction Module. The feature extraction module is utilized to extract features from noisy images and output the data tokens to guide the generation of the image function for the denoised images. We mainly follow DiT Peebles & Xie (2023) and deliver adaLN-Zero Transformer blocks to form a feature extraction encoder. As presented in Figure 5 (a), we adopt a linear layer to regress the dimension-wise scale and shift parameters α, γ, and τ from noise level embedding.

Function Prediction Module. The function prediction module is designed for generating the INR parameters β for the image function based on the image feature extracted from the feature extraction

Table 1: Table for the one-step image generation performance. The number in the name means the training resolution.

Dataset	Models	FID (↓)	SFID (↓)	IS (↑)	P (↑)	R (↑)
Cifar10-32	CM-UNet	32.87	20.94	6.16	**0.595**	0.23
	CM-Func	**28.87**	**19.81**	**6.92**	0.52	**0.30**
CelebA-64	CM-UNet	54.41	72.86	1.77	0.43	0.021
	CM-Func	**29.49**	**45.10**	**2.08**	**0.78**	**0.094**
CelebA-128	CM-UNet	89.46	158.64	1.40	0.46	0
	CM-Func	**69.3**	**124.22**	**1.66**	**0.46**	**0.002**
LSUN Church-128	CM-UNet	58.33	88.33	1.82	0.31	0.007
	CM-Func	**34.94**	**86.28**	**2.42**	**0.33**	**0.053**
LSUN Classroom-128	CM-UNet	65.18	84.69	2.54	0.41	0.026
	CM-Func	**57.96**	**76.31**	**2.83**	**0.50**	**0.033**

module. Before training, we randomly initialize the learnable INR tokens according to the shape of the parameters β. As shown in Figure 5 (b), we design a decoder block that adopts multi-head cross attention to fuse the data feature with INR tokens and predict the INR parameters for the image function of specific denoised images. Note that we follow Chen *et al.* Chen & Wang (2022) to use a grouping strategy to improve the efficiency and scalability of our function prediction module.

Render Module. After obtaining INR parameters β for the image functions, we need to differentially render images to ensure the whole pipeline is differential. As discussed before, we form a continuous image function as an MLP with parameter β corresponding to a specific image. With given resolution $H \times W$, we sample a coordinate list $\{(\frac{i}{H}, \frac{j}{W})\}_{i,j}$ where $i \in [0, H)$ and $j \in [0, W)$ and query the RGB value at each coordinate, which finally can be reshaped to form a complete image. There is no learnable parameter in the render module since the INR parameter β is generated by the function prediction module.

3.4 Benefit from Image Reconstruction Pre-training

We notice that predicting the image function for a clean image from a noisy image is pretty difficult if not impossible, due to the large search space of the INR parameter β. Therefore, we propose an image reconstruction task to pre-train the function prediction module. We hope the feature extraction module focuses on transforming the noise image feature into the clean image feature while the function prediction module focuses on transforming the clean image feature into its image function.

Therefore, we design an image reconstruction pre-training task for the function prediction module. as shown in Figure 5 (c). Specifically, during the image reconstruction pre-training task, we skip the feature extraction module and set the noise level σ as 0 so that the input image are neither perturbed with noise nor denoised by the feature extraction module. The pipeline is downgraded to a model (only the function prediction module contains learnable parameters) that transforms an input clean image to its corresponding image function with specific INR parameters β. We then optimize such a model with image reconstruction loss:

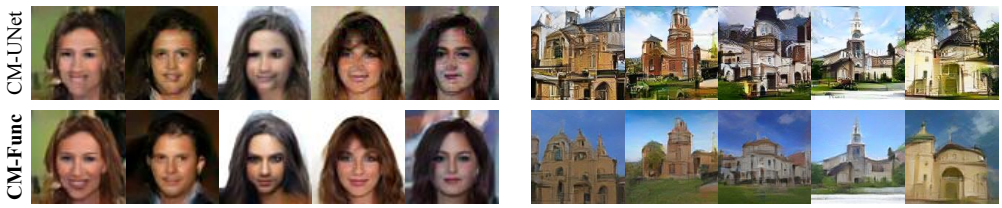
$$\mathcal{L}_{\text{rec}} = \texttt{MSE}(M_{\boldsymbol{G}_\theta(\mathbf{x}, \sigma=0)}\left(\{\mathrm{c}_i\}\right), \mathbf{x}), \tag{10}$$

where MSE denotes the mean square error loss.

After pre-training the function prediction module, we train the whole model (including the feature extraction module and function prediction module) with the consistency training objective shown in Eq. 5, which enables our model to generate new images with given random noise. We will show that the image reconstruction pre-training task helps to make the consistency training process converge faster.

4 Experiments

In the experiment section, we first provide a detailed comparison between the original U-Net generator and our function generator in the one-step image generation task based on consistency models under the low-training resource, denoted as *CM-UNet* and *CM-Func* respectively. Then we provide



(a) CelebA-64 (b) Church Outdoor-128

Figure 6: Results from (a) the CelebA dataset with resolution 64 (a) and from (b) the LSUN Church Outdoor with resolution 128. All corresponding images are generated from the same initial noise.

more discussion and ablation studies to illustrate the advantages of function generator when sampling images with different resolutions compared to other popular generators, such as UViT Bao et al. (2023a) and DiT Peebles & Xie (2023).

4.1 Setting

Datasets. We show the performance for the unconditional image generation on two popular image datasets: CelebA Liu et al. (2015) with resolution 64 and 128 and LSUN dataset Yu et al. (2015) with resolution 128. We also show that our pipeline can be applied to the class-conditional image generation with the Cifar10 dataset Krizhevsky et al. (2009). We mainly evaluate the models by sampling images with corresponding resolutions.

Hyper-parameters. Following the setting of training the consistency models as in Song *et al.* Song et al. (2023), we use the following default hyper-parameters for all datasets unless otherwise stated: $\sigma_{\min} = 0.002$, $\sigma_{\max} = 80$. We use a low batch-size training strategy to evaluate the performance of consistency models under the low-training resource setting. We set the number of encoder blocks N = 8 and the number of decoder blocks M = 6. We adopt MLP with a depth of 5 and width of 256, with ReLU activation, and positional embedding as our image function. For detailed hyper-parameters for consistency models and optimization processes, we present more details in Appendix A.

For the pre-training task, we use the same dataset as the consistency training task and follow Chen *et al.* Chen & Wang (2022) to optimize the models with Adam optimizer Kingma & Ba (2014) and learning rate $1e-4$ for 30 epochs. All results are reported for models with the pre-training task unless otherwise discussed.

Metric. We first compare the efficiency of consistency models with the U-Net generator and function generator. Then we report the quantitative generation results according to Frechet Inception Distance (FID) Heusel et al. (2017), Sliding Fréchet Inception Distance (sFID) Szegedy et al. (2016), Inception Score (IS) Salimans et al. (2016), Precision (P) Kynkäänniemi et al. (2019), and Recall (R) Kynkäänniemi et al. (2019). We follow Song et al. (2023) and Dhariwal & Nichol (2021) to generate 50000 images for credible scores.

We also define a new metric, the total denoising distance, to reflect the denoising quality of our generators during training. The total denoising distance is simply calculated as the L2 difference between the denoised result from the noisy image and the original clean image:

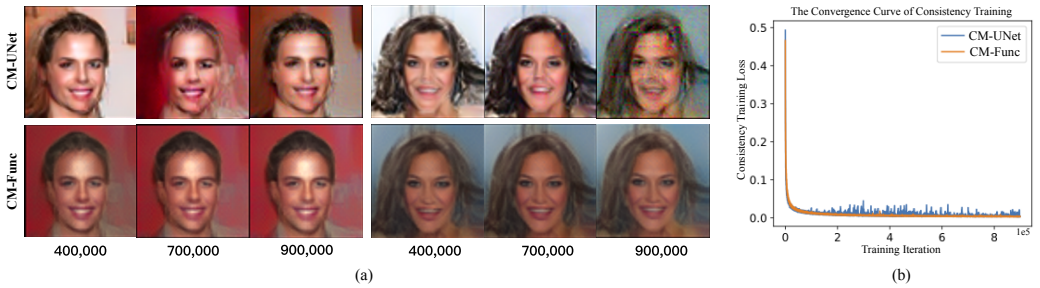
$$d^i_{\text{denoising}} = |\boldsymbol{f}_\theta\left(\mathbf{x}_{\sigma_i}, \sigma_i\right) - \mathbf{x}_0|.$$

Though this metric does not reflect the generation performance in the test phase, it means the denoising ability of the generator during training. More details for this metric are in Appendix B.

4.2 Results

Model efficiency. We quantitatively compare the efficiency of our function generator and the baseline U-Net generator on the unconditional image generation task to show that our model is more efficient. We report the total parameters, the GPU cost, and the inference FPS in Figure 2. We verify that our model has fewer total parameters, less training GPU cost, and higher inference FPS.

Apart from the absolute quantitative value, we also observe that when increasing the resolution of images (from Cifar10-32 to CelebA-64 and then to CelebA-128), the increment of the total param-



(a) (b)

Figure 7: (a) Visual results are generated from the two initial noises by models with different optimization iterations. (b) The convergence curve for the consistency training. The results indicate that the U-Net is unstable while our model is much more stable when keeping training.

Models	FID (↓)	sFID (↓)	IS (↑)	P (↑)
w. pretraining	**42.78**	**56.66**	**1.83**	**0.70**
w.o pretraining	66.61	107.72	1.81	0.37

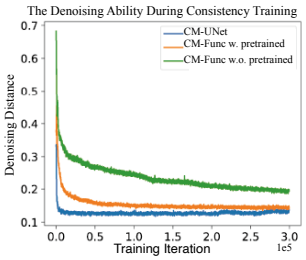


Figure 8: Top Left: Visual results of optimizing models with or without pre-training task for 50,000 iterations. Bottle Left: Alabtion of Pre-training task on CelebA-64 with 400,000 training iterations. Right: Denoising ability during training. Pre-training task greatly improves the denoising ability.

eter and GPU cost of our model is much less than the U-Net or even negligible, which verifies that our model has better scalability than U-Net when scaling to image with higher resolution.

Generation performance. We demonstrate the quantitative generation performance in Table 1. We can observe that the generation performance on the Cifar10, CelebA, and LSUN datasets beats the baseline U-Net in terms of all evaluation metrics, while our model has much fewer model parameters and enjoys faster training, faster inference, and flexible employment.

We also present some visual generation results based on the CelebA dataset with resolution 64 and the LSUN dataset with resolution 128 in Figure 6. We can clearly observe that given exactly the same initial noise, our model can generate much more realistic than the baseline consistency model with the U-Net generator. More visual and quantitative results are presented in Appendix D.

4.3 Discussions

Training oscillates for Consistency models with U-Net generator. We go deeper into the failing case when training consistency models with U-Net and low batch size 32 and show the results in Figure 7. We find that during the training of the consistency model with U-Net with a low batch size (32 for one single A6000 GPU), the generation results tend to be corrupted and fail to be denoised. On the contrary, the results from the consistency model with our function generator are much more consistent during training and we can find that these results are clean. We also present the consistency training objective convergence curve for the training consistency model in Figure 7 (b). The result shows that the consistency training process with the U-Net generator is pretty unstable and oscillating. In contrast, the process with our function generator is much more stable and consistent.

Pre-training improves the convergence of the consistency training with the function generator. We compare the total denoising distance metric of training consistency models with our function generator with or without pre-training task and show the result in Figure 8. We also show the generation performance at 400,000 iterations on the CelebA-64 dataset by ablating the pre-training task in the Bottle Left Table in Figure 8. The results show that our pre-training reconstruction task greatly improves the denoising ability of our function generator during the early stage of consistency training and leads to better generation performance. We also verify that our pre-training task is very efficient and costs much less time compared with the consistency training process. More quantitative results for the efficiency of the pre-training task are shown in Appendix C.

Table 2: Table for efficiency and accuracy of different generators on CelebA-64 dataset.

Models	Efficience		Accuracy			
	Any-Resolution Sampling	Multi-Resolution Sampling FPS (↑)	FID (↓)	sFID (↓)	IS (↑)	P (↑)
CM-UNet	✗	83.64	54.41	72.86	1.77	0.43
CM-UViT	✗	83.58	40.14	41.47	1.90	0.68
CM-DiT	✗	96.61	25.52	37.27	2.00	0.81
CM-Func	✓	**447.02**	29.49	45.10	2.08	0.78

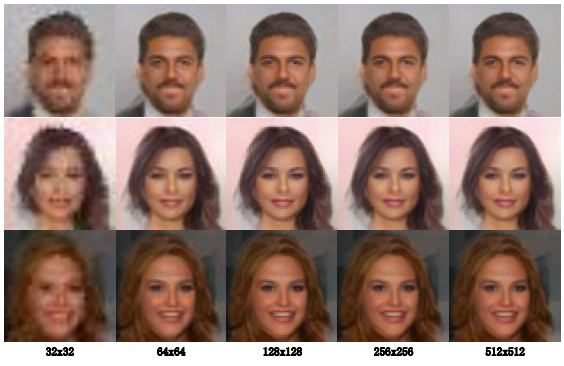


Figure 9: After training on the CelebA-64 dataset, our pipeline supports sampling at any resolution while the results at all resolutions remain clean and realistic.

Flexible sampling image resolution leads to a more efficient sampling process. A comprehensive experiment on the CelebA dataset shown in Table 2 is conducted to evaluate the efficiency and generation quality. We evaluate the efficiency by measuring the FPS to sample 10000 image signals, each of which requires 3 resolutions ($32 \times 32, 64 \times 64, 128 \times 128$, totally 30000 images). Only our function generator supports sampling any-resolution images with one model, while other generators require training multiple separate models at each resolution. Therefore, the multi-resolution sample FPS for those generators that generates fixed-resolution images is calculated as $\frac{30000}{\sum_{i=1}^{10000} T_{32}^i + \sum_{i=1}^{10000} T_{64}^i + \sum_{i=1}^{10000} T_{128}^i}$, where T_k^i is the average time that generates i^{th} image with resolution k. Since the function generator only needs to generate one image function for each image, the multi-resolution sample FPS for our function generator is calculated as $\frac{30000}{\sum_{i=1}^{10000} (T^i + T_{R32} + T_{R64} + T_{R128})}$, where T^i is the average time that generates i^{th} image functions, and $T_{R32}, T_{R64}, T_{R128}$ are the time to render the image function to image with 32/64/128 resolutions (which is nearly negligible compared to T^i). The results indicate that our function generator achieves a much higher multi-resolution sampling FPS. In contrast, other generator needs to deliver different models to separately denoise input noisy images, which leads to a slower sampling process.

For generation quality, we find that our function generator greatly beats U-Net and is comparable with Transformer-based generators. We show the generation results with different sampling resolutions in Figure 9. The results indicate that the sampling results at all resolutions remain clean and realistic. We also find that even trained with lower-resolution images, our function generator is still better than the U-Net generator. In addition, we find that our image functions have a better interpolation performance than the linear interpolation. See more quantitative and visualization results in Appendix D.2 and Appendix D.3.

5 Conclusion

In this paper, we explore efficient consistency training generation for consistency models. We observe that the U-Net generator is resource-intensive and inflexible. Reducing batch size for single GPU use harms performance. To address this, we introduce a Transformer generator that generates image functions parameterized as INR and then renders them into images of any resolution. We design a new end-to-end function predictor that simplifies generating clean image functions from noisy images. Additionally, we enhance training efficiency with an image reconstruction pre-training task. Extensive experiments show our method produces more realistic images with fewer resources than the original consistency models and also provides an efficient any-resolution image sampling process. A limitation of this paper lies in that INRs for image functions are global representations of signals, and have poor representation ability of the local semantic information, which makes our pipeline hard for generation with finer details.

REFERENCES

Fan Bao, Shen Nie, Kaiwen Xue, Yue Cao, Chongxuan Li, Hang Su, and Jun Zhu. All are worth words: A vit backbone for diffusion models. In *CVPR*, pp. 22669–22679, June 2023a.

Fan Bao, Shen Nie, Kaiwen Xue, Chongxuan Li, Shi Pu, Yaole Wang, Gang Yue, Yue Cao, Hang Su, and Jun Zhu. One transformer fits all distributions in multi-modal diffusion at scale. *arXiv preprint arXiv:2303.06555*, 2023b.

Lucy Chai, Michael Gharbi, Eli Shechtman, Phillip Isola, and Richard Zhang. Any-resolution training for high-resolution image synthesis. In *European Conference on Computer Vision*, pp. 170–188. Springer, 2022.

Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, and William Chan. Wavegrad: Estimating gradients for waveform generation. *arXiv preprint arXiv:2009.00713*, 2020.

Yinbo Chen and Xiaolong Wang. Transformers as meta-learners for implicit neural representations. In *ECCV*, pp. 170–187. Springer, 2022.

Yinbo Chen, Oliver Wang, Richard Zhang, Eli Shechtman, Xiaolong Wang, and Michael Gharbi. Image neural field diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8007–8017, 2024.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *NIPS*, 34: 8780–8794, 2021.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Emilien Dupont, Hyunjik Kim, SM Eslami, Danilo Rezende, and Dan Rosenbaum. From data to functa: Your data point is a function and you can treat it like one. *arXiv preprint arXiv:2201.12204*, 2022.

Ziya Erkoç, Fangchang Ma, Qi Shan, Matthias Nießner, and Angela Dai. Hyperdiffusion: Generating implicit neural fields with weight-space diffusion. *arXiv preprint arXiv:2303.17015*, 2023.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, pp. 1126–1135. PMLR, 2017.

Elizabeth Fons, Alejandro Sztrajman, Yousef El-Laham, Alexandros Iosifidis, and Svitlana Vyetrenko. Hypertime: Implicit neural representation for time series. *arXiv preprint arXiv:2208.05836*, 2022.

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.

Moayed Haji-Ali, Guha Balakrishnan, and Vicente Ordonez. Elasticdiffusion: Training-free arbitrary size image generation, 2023.

William Harvey, Saeid Naderiparizi, Vaden Masrani, Christian Weilbach, and Frank Wood. Flexible diffusion modeling of long videos. *NIPS*, 35:27953–27965, 2022.

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. *NIPS*, 30, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NIPS*, 33: 6840–6851, 2020.

Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.

Animesh Karnewar, Andrea Vedaldi, David Novotny, and Niloy J Mitra. Holodiffusion: Training a 3d diffusion model using 2d images. In *CVPR*, pp. 18423–18433, 2023.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *NIPS*, 35:26565–26577, 2022.

Junjie Ke, Qifei Wang, Yilin Wang, Peyman Milanfar, and Feng Yang. Musiq: Multi-scale image quality transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 5148–5157, 2021.

Chiheon Kim, Doyup Lee, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Generalizable implicit neural representations via instance pattern composers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11808–11817, 2023.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *NIPS*, 32, 2019.

Chieh Hubert Lin, Hsin-Ying Lee, Yen-Chi Cheng, Sergey Tulyakov, and Ming-Hsuan Yang. Infinitygan: Towards infinite-pixel image synthesis. In *International Conference on Learning Representations*, 2022.

Ke Liu, Feng Liu, Haishuai Wang, Ning Ma, Jiajun Bu, and Bo Han. Partition speeds up learning implicit neural representations based on exponential-increase hypothesis. In *ICCV*, pp. 5474–5483, 2023a.

Ke Liu, Ning Ma, Zhihua Wang, Jingjun Gu, Jiajun Bu, and Haishuai Wang. Implicit neural distance optimization for mesh neural subdivision. In *ICME*, pp. 2039–2044, 2023b. doi: 10.1109/ICME55011.2023.00349.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Xingchao Liu, Xiwen Zhang, Jianzhu Ma, Jian Peng, and Qiang Liu. Instaflow: One step is enough for high-quality diffusion-based text-to-image generation. *arXiv preprint arXiv:2309.06380*, 2023c.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, pp. 3730–3738, 2015.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *NIPS*, 35:5775–5787, 2022a.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022b.

Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378*, 2023.

Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *CVPR*, pp. 14297–14306, 2023.

Anish Mittal, Rajiv Soundararajan, and Alan C. Bovik. Making a "completely blind" image quality analyzer. *IEEE Signal Processing Letters*, 20(3):209–212, 2013. doi: 10.1109/LSP.2012.2227726.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, pp. 165–174, 2019.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, pp. 4195–4205, 2023.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *ICML*, pp. 5301–5310. PMLR, 2019.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, pp. 10684–10695, 2022.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pp. 234–241. Springer, 2015.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *NIPS*, 29, 2016.

Vincent Sitzmann, Eric Chan, Richard Tucker, Noah Snavely, and Gordon Wetzstein. Metasdf: Meta-learning signed distance functions. *NIPS*, 33:10136–10147, 2020a.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *NIPS*, 33:7462–7473, 2020b.

Ivan Skorokhodov, Savva Ignatyev, and Mohamed Elhoseiny. Adversarial generation of continuous images. In *CVPR*, pp. 10753–10764, 2021.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, pp. 2256–2265. PMLR, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020a.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *NIPS*, 32, 2019.

Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *NIPS*, 33:12438–12448, 2020.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020b.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.

Filip Szatkowski, Karol J Piczak, Przemysław Spurek, Jacek Tabor, and Tomasz Trzciński. Hypernetworks build implicit neural representations of sounds. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, pp. 2818–2826, 2016.

Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P Srinivasan, Jonathan T Barron, and Ren Ng. Learned initializations for optimizing coordinate-based neural representations. In *CVPR*, pp. 2846–2855, 2021.

Jianyi Wang, Kelvin CK Chan, and Chen Change Loy. Exploring clip for assessing the look and feel of images. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 2555–2563, 2023.

Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. In *Computer Graphics Forum*, volume 41, pp. 641–676. Wiley Online Library, 2022.

Sidi Yang, Tianhe Wu, Shuwei Shi, Shanshan Lao, Yuan Gong, Mingdeng Cao, Jiahao Wang, and Yujiu Yang. Maniqa: Multi-dimension attention network for no-reference image quality assessment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1191–1200, 2022.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6613–6623, 2024.

Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, pp. 586–595, 2018.

Shen Zhang, Zhaowei Chen, Zhenyu Zhao, Zhenyuan Chen, Yao Tang, Yuhao Chen, Wengang Cao, and Jiajun Liang. Hidiffusion: Unlocking high-resolution creativity and efficiency in low-resolution trained diffusion models. *arXiv preprint arXiv:2311.17528*, 2023.

Shuyi Zhang, Ke Liu, Jingjun Gu, Xiaoxu Cai, Zhihua Wang, Jiajun Bu, and Haishuai Wang. Attention beats linear for fast implicit neural representation generation. *arXiv preprint arXiv:2407.15355*, 2024.

Peiye Zhuang, Samira Abnar, Jiatao Gu, Alex Schwing, Joshua M Susskind, and Miguel Angel Bautista. Diffusion probabilistic fields. In *The Eleventh International Conference on Learning Representations*, 2023.

A Detailed Experiment Settings

In this section, we first present the detailed hyper-parameters that are used to train the consistency models (Section A.1) and the optimization processes (Section A.2). And then we show the detailed architecture of the U-Net that we compare in the main experiments (Section A.3).

A.1 More hyper-parameters in consistency models

In this part, we present more details about the hyper-parameters in consistency models Song et al. (2023). We mainly follow the original setting by consistency models Song et al. (2023).

The hyper-parameters in Eq. 2 that parameterized the boundary condition are defined as:

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

where $\sigma_{\text{data}} = 0.5$ and $\sigma_{\min} = 0.002$.

In our experiments, we utilize the LPIPS metric Zhang et al. (2018) to compose $d(\cdot, \cdot)$ as in Eq. 4 and Eq. 5 while interpolating the images to resolution 224×224. The noise level $\sigma_i \in [\sigma_{\min}, \sigma_{\max}]$ are discretized into N sub-intervals with the following equation:

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

where s_0, s_1 are the minimal or maximal number of sub-intervals, while k and K are the current and total training step. For all unconditional image generation tasks, $s_0 = 2$ and $s_1 = 150$. Following Karras *et al.* Karras et al. (2022), the exact value of σ_i is defined as:

$$\sigma_i = \left(\sigma_{\min}^{\frac{1}{\rho}} + \frac{i-1}{N(k)-1}\left(\sigma_{\max}^{\frac{1}{\rho}} - \sigma_{\min}^{\frac{1}{\rho}}\right)\right)^{\rho},$$

where $i \in [1, N(k)]$ and $\rho = 7$.

A.2 More training details of our experiments

Unless otherwise stated, we follow Peebles *et al.* Peebles & Xie (2023) and Chen *et al.* Chen & Wang (2022) to split the 32×32 and 64×64 images with patch size 4, and the 128×128 images with patch size 8. We set up the number of encoder block $N = 8$ and the number of decoder block $M = 6$. All of the models are optimized with Rectified Adam optimizer Liu et al. (2019) under the learning rate $5e-4$.

Due to the limitation of training resources, the models for unconditional image generation tasks are optimized with a batch size of 32 on a single 48 GB A6000 GPU. We optimize the models on the Cifar-10 dataset for 400,000 iterations and the models on the CelebA dataset and LSUN datasets for 900,000 iterations to ensure the models converge.

A.3 Detailed Architecture of the U-Net

In this part, we present the detailed architecture of the baseline U-Net. Specifically, we follow the U-Net architecture as in Song *et al.* Song et al. (2023) and Dhariwal *et al.* Dhariwal & Nichol (2021). The detailed architecture hyper-parameters are defined as follows:

- Attention is applied at three different resolutions, including 32×32, 16×16, and 8×8.
- The time embedding is generated with a learnable MLP that is composed of two linear layers whose embedding dimension is 1024. The activation function in the MLP is SiLU. Then the time embedding works as an input for every layer of the U-Net.

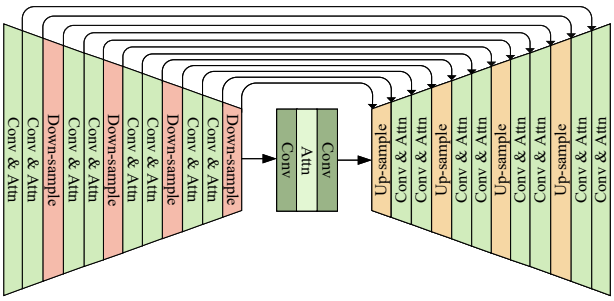


Figure 10: The detailed architecture of baseline U-Net. It contains heavy convolution and attention at different resolutions. The input contains the input noisy images and the time embeddings.

- The channel multiplication has four levels: [1,2,3,4], while in each level, we have 2 resident blocks for the unconditional image generation tasks.
- The dimension of the channel in the resident blocks is set to 256.
- For the unconditional image generation task, we do not use scale and shift norm in the residual blocks.
- The number of the attention head in both down-sample blocks and up-sample blocks is set to 4, while the channel of the head is set to 64.
- Dropout is not applied in our setting.
- The detailed architecture is shown in Figure 10.

B THE TOTAL DENOISING DISTANCE METRIC

Note that we follow Eq. 4 to enforce the denoised results of $\mathbf{x}_{\sigma_{i+1}}$ and $\mathbf{x}_{\sigma_i}$ to be close with consistency training objective:

$$d_{\mathrm{CT}}^i = \boldsymbol{f}_\theta\left(\mathbf{x}_{\sigma_{i+1}}, \sigma_{i+1}\right) - \boldsymbol{f}_\theta\left(\mathbf{x}_{\sigma_i}, \sigma_i\right).$$

We can achieve good image denoising only if d_{CT}^i is small for all $\mathbf{x}_{\sigma_i}$ in the PF-ODE trajectory and we have $\boldsymbol{f}_\theta\left(\mathbf{x}_{\sigma_i}, \sigma_i\right) \approx \mathbf{x}_0$. Therefore, we can define a total denoising distance to reflect the quality along PF-ODE trajectory of our denoising model during training:

$$d_{\mathrm{denoising}}^i = |\sum_{j=0}^{i} d_{\mathrm{CT}}^j| = |\boldsymbol{f}_\theta\left(\mathbf{x}_{\sigma_i}, \sigma_i\right) - \mathbf{x}_0|,$$

which can be simply calculated as the difference between the denoised image and the original clean image. We can use this metric to monitor the denoising ability of the consistency model during training.

C TRAINING EFFICIENCY ANALYSIS

We evaluate the total training time for different models on the celebA-64 dataset to show the efficiency of our model pipeline and training mechanism. As shown in Table 3, the time consumed by the pre-training tasks only accounts for 5% of the total training time. This is because 1) the pre-training task for reconstruction needs much fewer iterations to converge, 2) the pre-training is applied solely to the Function Prediction Module, which accounts for less than half of the total model parameters. Therefore, our pre-training task is very efficient and the extra training cost is almost negligible compared with the heavy cost for consistency training.

Table 3: Table for training efficiency on CelebA-64 dataset.

	Total Training Time (GPU * Hours) for CelebA-64
CM-UNet	302.5
CM-DiT	207.5
CM-UViT	212.5
CM-Func w.o. pre-training	223.75
CM-Func w. pre-training	223.75 + 12.9

Table 4: Table for the generation performance on Cifar10-32 and CelebA-64 dataset in terms of image quality assessment metrics.

	Models	NIQE (↓)	CLIPIQA (↑)	MUSIQ (↑)	MANIQA (↑)
Cifar10-32	CM-UNet	23.003	0.516	17.043	0.105
	CM-Func	**22.264**	**0.520**	**17.143**	**0.107**
CelebA-64	CM-UNet	6.810	0.465	21.198	0.210
	CM-Func	**6.503**	**0.549**	**22.230**	**0.220**

D More Experiment Results

In this part, we present more experiment results to illustrate the performance of our function generator compared to the U-Net generator (Section D.1). We also adapt two more Transformer-based generators that were originally used in diffusion models to the consistency model pipeline and compare the performance between the Transformer-based generators and the U-Net generator (Section D.2). Finally, we show that our function generator enjoys the advantage of generating images with user-specific resolutions compared with the classical Transformer-based generators (Section D.3).

D.1 More Results

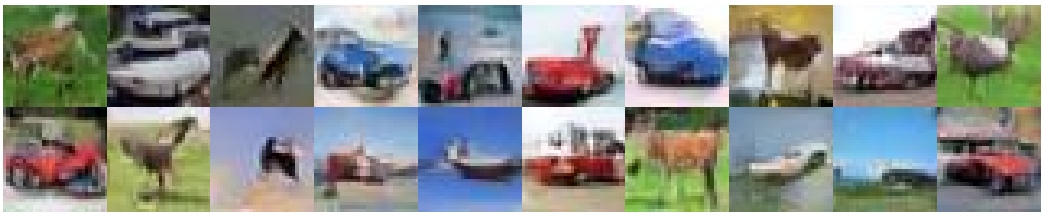
We include more quantitative metrics for no-reference image quality assessment to evaluate the performance of our function generator and the baseline U-Net. Specifically, we use the pyiqa package to evaluate these four metrics, i.e., NIQE Mittal et al. (2013), CLIPIQA Wang et al. (2023), MUSIQ Ke et al. (2021), and MANIQA Yang et al. (2022). We follow the default setting to evaluate NIQE, CLIPIQA, and MUSIQ on 50000 generated images and evaluate MANIQA on 1000 generated images for efficiency. The results are presented in Table 4, which demonstrates that the quality of the images generated by our function generator is better than those by the U-Net generator.

We also present the visual results for consistency models trained on Cifar10-32 dataset Krizhevsky et al. (2009) in Figure 11, the visual results for consistency models trained on CelebA-64 dataset Liu et al. (2015) in Figure 12, the visual results for consistency models trained on CelebA-128 dataset Liu et al. (2015) in Figure 13, the visual results for consistency models trained on LSUN-128 dataset Yu et al. (2015) in Figure 14 and Figure 15.

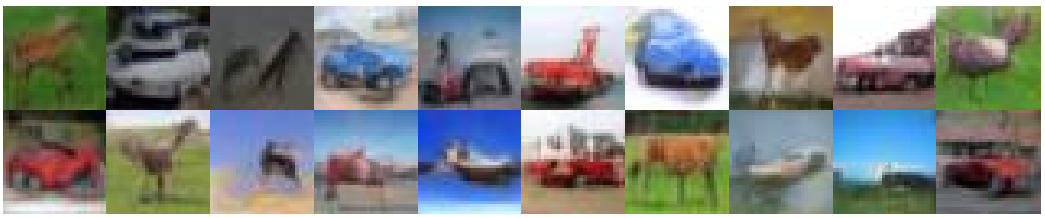
On the Cifar10 dataset, the visual results are not so good. We owe this problem to the limited supervised signals of the Cifar10 dataset. The Cifar10 dataset contains images with low resolution (32×32) and can provide very limited supervised signals to predict good image functions. However, even though the visual generation results are not so good, we can observe the difference between the results from the two generators, while the results from the U-Net are more broken and contain more noise. On the contrary, the results from our image generator tend to be cleaner and more vivid. These results show that our image generator performs better than the U-Net generator even on images with pretty low resolution.

On the CelebA-64 and the CelebA-128 datasets, as presented in Figure 12 and Figure 13, we can observe a more obvious broken case of the U-Net generator, which further confirms our conclusion. On the contrary, the consistency model with our function generator tends to generate more clear, realistic, diverse images.

For the LSUN Church-128 datasets, as presented in Figure 14, we find that the U-Net generator tends to generate more artifacts, which prevents the U-Net from generating clearer and more realistic



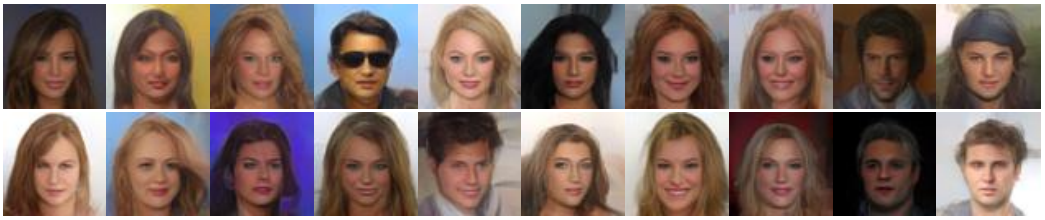
(a) Results from U-Net generator on Cifar10-32 dataset



(b) Results from function generator on Cifar10-32 dataset

Figure 11: Visual results from U-Net generator (a) and function generator (b) trained from the Cifar10-32 dataset, while images in the same location are generated from the same initiated noises. Even though these visual generation results are not so good, we can find the difference between the results from the two generators, while the results from the U-Net are more broken and contain more noise. On the contrary, the results from our function generator tend to be cleaner and more vivid.

(a) Results from U-Net generator on CelebA-64 dataset



(b) Results from function generator on CelebA-64 dataset

Figure 12: Visual results from U-Net generator (a) and function generator (b) trained from the CelebA-64 dataset, while images in the same location are generated from the same initiated noises. We can observe that the results from the U-Net generator are still noisy. On the contrary, the consistency model with our function generator tends to generate more clear, realistic, diverse images.

images. For the LSUN Classroom-128 datasets, we show the visual results in Figure 15. On such a complex scene dataset, we find that the U-Net also generates more meaningless artifacts. Even though our function generator does not generate very clear details due to the low training batch, it at least generates rough but clear images.

(a) Results from U-Net generator on CelebA-128 dataset

(b) Results from function generator on CelebA-128 dataset

Figure 13: Visual results from U-Net generator (a) and function generator (b) trained from the CelebA-128 dataset, while images in the same location are generated from the same initiated noises. We can observe that the results from the U-Net generator are broken due to the training oscillation. On the contrary, the consistency model with our function generator tends to generate more clear, realistic, diverse images.

(a) Results from U-Net generator on LSUN Church-128 dataset

(b) Results from function generator on LSUN Church-128 dataset

Figure 14: The visual results from U-Net generator (a) and function generator (b) trained from the LSUN Church-128 dataset, while images in the same location are generated from the same initiated noises. We can observe that the results from the U-Net generator contain lots of artifacts. On the contrary, the consistency model with our function generator tends to generate more clear, realistic, diverse images.

(a) Results from U-Net generator on LSUN Classroom-128 dataset

(b) Results from function generator on LSUN Classroom-128 dataset

Figure 15: The visual results from U-Net generator (a) and function generator (b) trained from the LSUN Classroom-128 dataset, while images in the same location are generated from the same initiated noises. We can observe that the results from the U-Net generator contain lots of artifacts. On the contrary, the consistency model with our function generator tends to generate more clear, realistic, diverse images.

D.2 Comparison with other generators

We adapt two more Transformer-based generators that were originally used in diffusion models to replace the U-Net generator in consistency models. One is U-ViT by Bao *et al.* Bao et al. (2023a) and the other one is DiT by Peebles *et al.* Peebles & Xie (2023).

D.2.1 Setting

We build these two generators that have a close amount of learnable parameters with our function generator. As a result, the parameter amount and GPU cost of DiT and UViT are very close to our function generators. The detailed architectures are presented as follows.

For U-ViT, the total depth is set to 15 so that the whole model contains 7 layers for the encoder, 1 layer for middle transformation, and 7 layers for the decoder. The skip connection is applied between the corresponding layers of the encoder and decoder. Same as our function generator, we set the patch size as 4 for 64×64 images and 8 for 128×128 images, the embedding dimension as 768, number of heads as 12. As for DiT, the same hyper-parameters (depth, patch size, embedding dimension, number of heads) are set to ensure they have a close amount of total parameters.

All models are optimized with the same iterations and the same optimizer as shown in Section 4.1. We evaluate the performance of Transformer-based generators and our function generator on the CelebA dataset with resolution 64×64.

We use extensive quantitative metrics to evaluate the performance of different generators, including FID Heusel et al. (2017), sFID Szegedy et al. (2016), IS Salimans et al. (2016), Precision Kynkäänniemi et al. (2019), Recall Kynkäänniemi et al. (2019), NIQE Mittal et al. (2013), CLIPIQA Wang et al. (2023), MUSIQ Ke et al. (2021), and MANIQA Yang et al. (2022). We follow the default setting to evaluate all metrics on 50000 generated images, except MANIQA which is evaluated on 1000 generated images for efficiency.

Table 5: Table for the generation performance of consistency models with different generators on CelebA-64 dataset. The best results are marked as bold and the second results are marked with underline.

Dataset	Models	FID (↓)	sFID (↓)	IS (↑)	precesion (↑)	Recall (↑)
CelebA-64	CM-UNet	54.41	72.86	1.77	0.43	0.021
	CM-UViT	40.14	41.47	1.90	0.68	0.020
	CM-DiT	**25.52**	**37.27**	2.00	**0.81**	**0.10**
	CM-Func	29.49	45.10	**2.08**	0.78	0.094

Table 6: Table for the generation performance for different generators on CelebA-64 dataset in terms of image quality assessment metrics. The best results are marked as bold and the second results are marked with underline.

	Models	NIQE (↓)	CLIPIQA (↑)	MUSIQ (↑)	MANIQA (↑)
CelebA-64	CM-UNet	6.810	0.465	21.198	0.210
	CM-UViT	6.564	0.516	**22.805**	0.194
	CM-DiT	6.666	**0.561**	22.139	**0.234**
	CM-Func	**6.503**	0.549	22.230	0.220

Table 7: Table for the generation performance of consistency models with different training resolutions. The results are evaluated on the CelebA dataset at the resolution of 128.

Models	Training Resolution	FID (↓)	sFID (↓)	IS (↑)	P (↑)	R(↑)
CM-UNet	128	89.46	158.64	1.40	0.46	0
CM-Func	128	69.30	124.22	1.66	0.43	0.002
	64	76.93	105.90	1.98	0.18	0.0011

D.2.2 RESULTS

Comparable with other Transformer-based generators. We present the quantitative results of consistency models with different generators in Table 2, Table 5 and and Table 6. We find that since our function generator is built based on Transformer architecture, it enjoys close generation performance with other Transformer-based generators. In the meanwhile, all generators based on Transformer have a better performance than the U-Net generator, which proves that Transformer as a basic architecture, is more stable and flexible than U-Net in the consistency model pipeline. Furthermore, compared with these two Transformer-based generators, our function generator enjoys the advantage of generating images with arbitrary resolution, which we will discuss in section D.3.

D.3 SUPPORT GENERATING IMAGES WITH USER-SPECIFIC RESOLUTIONS

D.3.1 MODIFICATION TO INFERENCE PHASE

With very little modification on the inference process, our pipeline supports one-step image generation with a user-specific resolution during the inference phase,

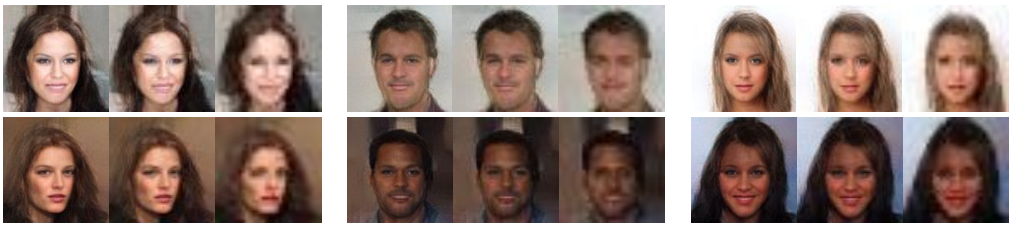
When we re-examine the inference process of consistency models with our function generator, we find that the main trouble that prevents our models from generating arbitrary-resolution images lies on Eq. 2:

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

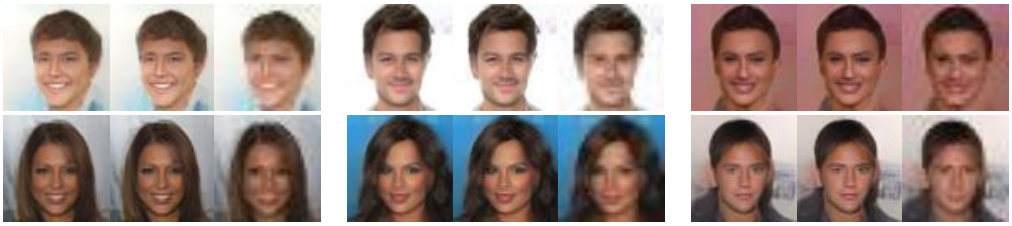
where the input noisy images $\mathbf{x}$ and the denoised images from the neural network generator $\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}, \sigma)$ must have the same resolution so that they can be added. However, during the inference phase, c_{skip} is a pretty small number which makes the input noisy images have little effect on the final results. Therefore, we simply modify the Eq. 2 by omitting the $c_{\text{skip}}(\sigma)\mathbf{x}$ term:

$$\boldsymbol{f}_{\boldsymbol{\theta}}(\mathbf{x}, \sigma) = c_{\text{out}}(\sigma)\boldsymbol{F}_{\boldsymbol{\theta}}(\mathbf{x}, \sigma).$$

Note that we completely de-couple the relationship between the resolution of the input noisy images and the output denoised images. The resolution of the images used to train the consistency models



(a) Sampling result examples from 128×128 Gaussian noise. From left to right, the sampling resolution is 128×128, 64×64,32×32



(b) Sampling result examples from 64×64 Gaussian noise. From left to right, the sampling resolution is 128×128, 64×64,32×32

Figure 16: The sampling result examples from 128×128 Gaussian noise (a) and 64×64 Gaussian noise (b). From left to right, the three images have resolution 128×128, 64×64, and 32×32. We can observe that the consistency model with our function generator can generate images with arbitrary resolution, while the images at all resolutions remain clean and realistic. We can also find that when sampling images with 128×128 resolution, the results from 128×128 Gaussian noise are more realistic than those from 64×64, which indicates that a larger noise space leads to better generation performance.

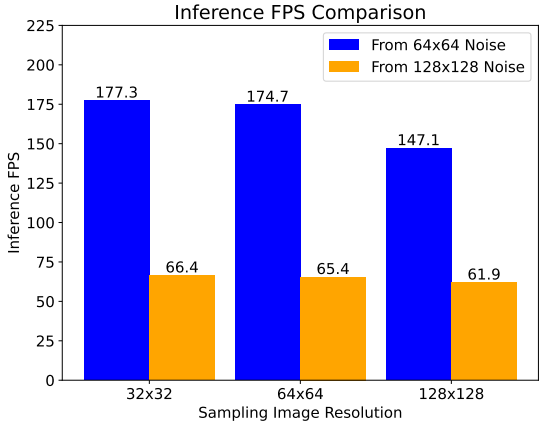


Figure 17: The inference FPS comparison of the consistency models inferring from 64×64 noise and 128×128 noise. The result shows that a smaller noise space leads to faster inference.

with our function generator only affects the noise space that we need to denoise. For example, training on 64×64 images indicates that we need to denoise 64×64 Gaussian noise vectors in the inference phase while training on 128×128 images indicates that we need to denoise 128×128 Gaussian noise vectors in the inference phase. A larger noise space typically implies more diverse image generation because more information can be extracted and stored in the neural network. On the contrary, a faster sampling speed can be obtained with a smaller noise space. We show the performance of our inference phase in Section D.3.2.

D.3.2 Results

Flexible sampling image resolution. We show that the consistency model with our function generator can generate images with arbitrary resolution in Figure 16. We can observe that the results of denoising from the noisy images at all resolutions remain clean and realistic. We can also find that when sampling images with 128×128 resolution, the results from 128×128 Gaussian noise are more

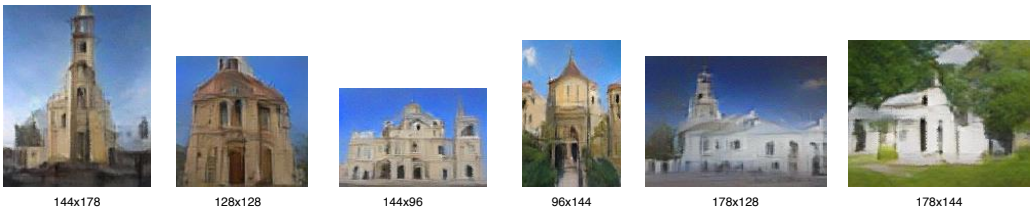


Figure 18: This figure shows some non-squared samples with different height-width ratios from the CM-Func model trained on the LSUN Church-128 dataset. By simply querying at different coordinates, we can achieve image generation with any height-width ratio.

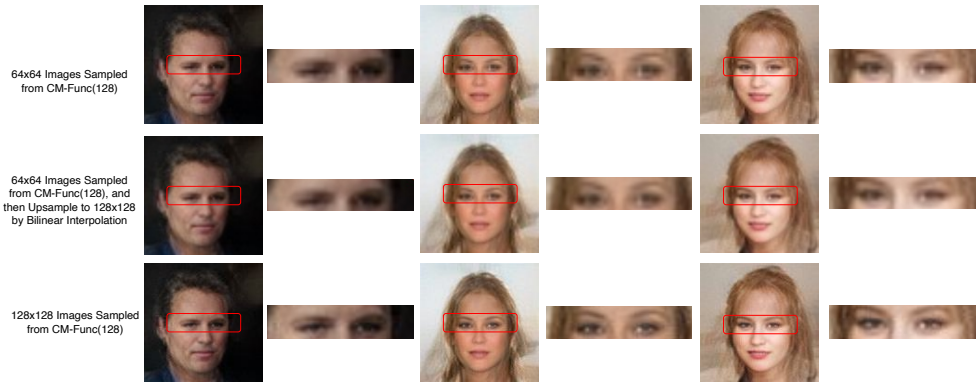


Figure 19: The visual comparison of our image function interpolation results and bi-linear interpolation results from CM-Func trained with 128x128-resolution images. We find that directly sampling 128x128 images from CM-Func has a better visual performance than sampling 64x64 images from CM-Func and then upsampling to 128x128 images by bi-linear interpolation. This result indicates that our CM-Func learns a more complex interpolation mechanism during training compared to the simple linear interpolation mechanism. **(Please see it in color version and zoom in for better visual performance)**

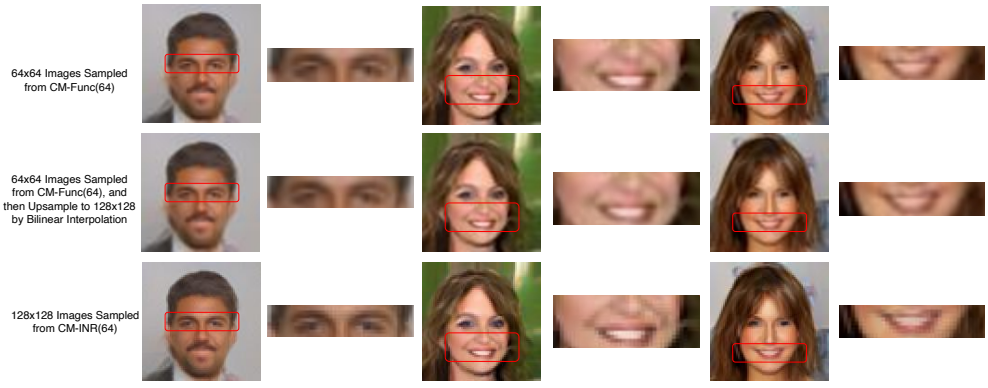


Figure 20: The visual comparison of our image function interpolation results and bi-linear interpolation results from CM-Func trained with 64x64-resolution images. We find that directly sampling 128x128 images from CM-Func (higher than training resolutions) leads to sharper edges than sampling 64x64 images from CM-Func and then upsampling to 128x128 images by bi-linear interpolation. This is because the image function based on INR is a complex continuous function that naturally supports more complex interpolation. **(Please see it in color version and zoom in for better visual performance)**

realistic than those from 64×64, which indicates that a larger noise space leads to better generation performance. We also present quantitative results on the CelebA-128 dataset with different training resolutions of our function generator in Table 7. We find that even trained with a lower resolution image, our function generator still has a good performance and is better than the U-Net generator. In addition, we present the inference FPS comparison in Figure 17, which shows that denoising from a smaller noise leads to faster inference speed. We also show that our function generator can generate non-squared samples with any height-weight ratio in Figure 18. This process can be easily implemented by querying the image function with coordinates of different height-width ratios.

Efficient sampling process for multi-resolution images. Our pipeline is the first work to generate functions instead of images. Based on such generated functions, we can decouple the resolution of generated images and the total amount of the parameters generated from the neural network. Our pipeline only needs to apply the inference forward once to get the image function, which can be rendered into images with any resolution. For example, if we use the traditional U-Net to generate 10000 images, each of which requires 3 different resolutions (32×32, 64×64 and 128×128), we need to separately apply the inference forward process with 3 different models on 30000 different noise images. On the contrary, our pipeline only needs to generate 10000 functions, each of which will be rendered into three images with negligible render time. As shown in Table 2, our pipeline achieves nearly about 400%+ FPS to sample all these images. Therefore, compared to the U-Net generators, our proposed function generator enjoys much better inference efficiency when generating images with different resolutions.

Better super-resolution performance than simple linear interpolation. We show the comparison of the interpolation performance of our image functions or simple linear interpolation in Figure 19 and Figure 20. In the case of sampling resolution no exceeding training resolution, we find that generating samples with 128x128 resolution directly with CM-Func-128 yields much better results than sampling at 64x64 and then upscaling to 128x128 with bilinear interpolation, as CM-Func learns some non-linear interpolation mechanism during optimization. In the case of sampling resolution exceeding training resolution, we find that generating 128x128 samples with CM-Func-64 also produces better visual results than sampling at 64x64 and then upscaling to 128x128 with bilinear interpolation, e.g. sharper edges, because INR inherently provides stronger non-linear interpolation capabilities.

E BROADER IMPACTS

Image generation is a widely discussed problem. Deep learning generative models can greatly impact society. Positively, they can enhance creative industries, healthcare, and education by generating art, improving medical imaging, and personalizing learning. However, they also pose risks, such as creating deepfakes, spreading misinformation, and raising ethical concerns about data privacy and bias. Balancing these benefits and risks is essential for their responsible use.