Bring Metric Functions into Diffusion Models

Jie An[1*], **Zhengyuan Yang**[2], **Jianfeng Wang**[2], **Linjie Li**[2],
Zicheng Liu[2], **Lijuan Wang**[2], **Jiebo Luo**[1]

[1]University of Rochester
[2]Microsoft
{jan6,jluo}@cs.rochester.edu, {zhengyang,jianfw,lindsey.li,zliu,lijuanw}@microsoft.com

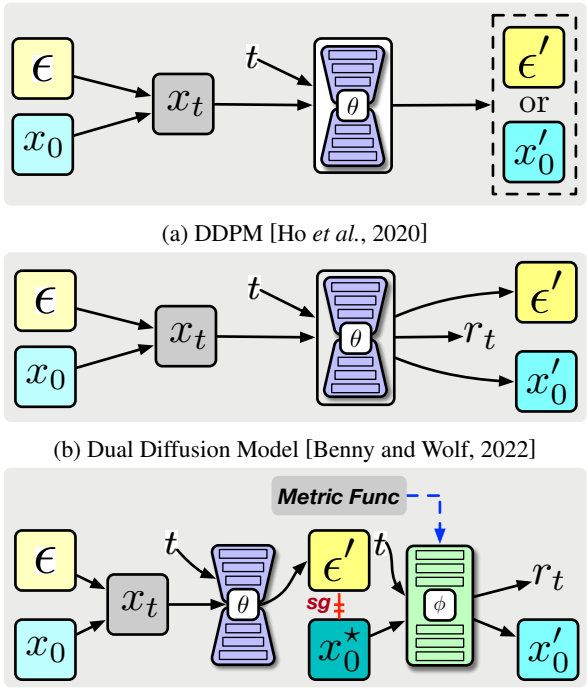
Abstract

We introduce a Cascaded Diffusion Model (**Cas-DM**) that improves a Denoising Diffusion Probabilistic Model (DDPM) by effectively incorporating additional metric functions in training. Metric functions such as the LPIPS loss have been proven highly effective in consistency models derived from the score matching. However, for the diffusion counterparts, the methodology and efficacy of adding extra metric functions remain unclear. One major challenge is the mismatch between the noise predicted by a DDPM at each step and the desired clean image that the metric function works well on. To address this problem, we propose Cas-DM, a network architecture that cascades two network modules to effectively apply metric functions to the diffusion model training. The first module, similar to a standard DDPM, learns to predict the added noise and is unaffected by the metric function. The second cascaded module learns to predict the clean image, thereby facilitating the metric function computation. Experiment results show that the proposed diffusion model backbone enables the effective use of the LPIPS loss, improving the image quality (FID, sFID) of diffusion models on various established benchmarks.

1 Introduction

The Denoising Diffusion Probabilistic Model (DDPM) [Ho *et al.*, 2020] has emerged as a leading method in visual content generation, positioned among other approaches such as Generative Adversarial Networks (GAN) [Goodfellow *et al.*, 2014], Variational Auto-Encoders (VAE) [Kingma and Welling, 2013], auto-regressive models [Esser *et al.*, 2021], and normalization flows [Kingma and Dhariwal, 2018]. DDPM is a score-based model that adopts an iterative Markov chain in generating images, where the transition of the chain is the reverse diffusion process to gradually denoise images.

Recently, [Song *et al.*, 2023] propose a novel score-based generative model called consistency model. One key observation is that using metric functions such as the Learned Perceptual Image Patch Similarity (LPIPS) loss [Zhang *et al.*, 2018] in training can significantly improve the quality of generated images. The LPIPS loss, with its VGG backbone [Simonyan and Zisserman, 2014] trained on the ImageNet dataset for classification, allows the model to capture more accurate and diverse semantic features, which may be hard to learn through generative model training alone. However, it remains unclear whether adding additional metric functions could yield similar improvements in diffusion models. In this study, taking



(a) DDPM [Ho *et al.*, 2020]

(b) Dual Diffusion Model [Benny and Wolf, 2022]

(c) Ours

Figure 1: We introduce a cascaded diffusion model that effectively incorporates metric functions in diffusion training. **(a)** DDPM outputs either ϵ' or x_0' and uses the corresponding loss in training. **(b)** Dual Diffusion Model outputs both ϵ' and x_0' simultaneously with a single network θ, where applying metric functions on x_0' will inevitably disturb the prediction of ϵ'. **(c)** Our Cas-DM cascades the main module θ with an extra network ϕ. θ is frozen for the x_0'-related losses and metric functions. The dashed blue line shows the gradient flow of the metric function. *sg* denotes stop gradient.

[*]Work done during internship at Microsoft.

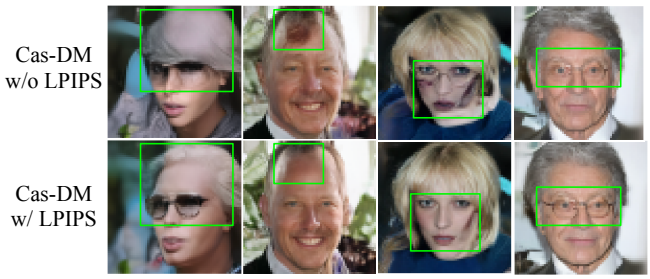


Figure 2: Qualitative comparison of Cas-DM [$\phi_{\texttt{UNet}}$] w/ and w/o LPIPS on CelebAHQ. Green boxes highlight differences in image details.

LPIPS loss as a prototype, we explore how to effectively incorporate metric functions into diffusion models. The primary challenge lies in the mismatch between the multi-step denoising process that generates noise predictions, and the single-step metric function computation that requires a clean image.

We next zoom in on the DDPM process to better illustrate this mismatch challenge. As shown in Figure 1, DDPM adopts a diffusion process to gradually add noise to a clean image x_0, producing a series of noisy images $x_i, i \in 1, ..., T$. Then the model is trained to perform a reverse denoising process by predicting a less noisy image x_{t-1} from x_t. Instead of directly predicting x_{t-1}, DDPM gives two ways to obtain x_{t-1}: predicting either the clean image x_0 or the added Gaussian noise ϵ_t. The training objective of the DDPM is the mean squared error (MSE) between the predicted and ground truth x_0 or ϵ_t, where a few papers [Nichol and Dhariwal, 2021; Benny and Wolf, 2022] found the latter (*i.e.*, the ϵ mode) to be empirically better than predicting x_0 (*i.e.*, the x_0 mode).

The two modes in DDPM provide us with two initial options for bringing metric functions. Applying metric functions directly to predicted noise ϵ is unreasonable because the networks for metric functions are trained on RGB images and produce meaningless signals when applied to noise ϵ. Despite the promising improvements, this x_0-mode model with metric functions still suffers from the low performance from the x_0-mode baseline, when compared with the ϵ-mode. This naturally motivates the question: can we merge the two modes and further improve the ϵ-mode performance with the metric function? To achieve this, we need a diffusion model that can generate x_0 while maintaining the ϵ-mode performance. The goal is made possible with the Dynamic Dual Diffusion Model [Benny and Wolf, 2022], where authors expand the output channel of the DDPM's network θ to let it predict x_0, ϵ, and a dynamic mixing weight, simultaneously. The experiments show that Dual Diffusion Model outperforms both the x_0- and ϵ-modes of DDPM. However, naively adding the metric function to its x_0 head will not work. This is because the additional metric functions on the predicted x_0 updates the shared backbone, which disturbs the ϵ prediction and leads to degraded performance.

To this end, we propose a new Cascaded Diffusion Model (**Cas-DM**), which allows the application of metric functions to DDPM by addressing the above-mentioned issues. We cascade two network modules, where the first model θ takes the noisy image x_t and predicts the added noise. We then derive an initial estimation of x_0 based on x_t and ϵ_θ following equations of the diffusion process. Next, the second model ϕ takes the initial x_0 prediction and the time step t and output the refined prediction of x_0 as well as the dynamic weight to mix x_0 and ϵ predictions in diffusion model sampling. In training, we apply the metric function to the predicted x_0 of ϕ, which is used to update the parameters of ϕ and stop the gradient for θ. This ensures the ϵ branch to be intact while the x_0 branch is enhanced by the additional metric function.

Experimental results on CIFAR10 [Krizhevsky *et al.*, 2009], CelebAHQ [Karras *et al.*, 2017], LSUN-Church/Bedroom [Yu *et al.*, 2015], and ImageNet [Deng *et al.*, 2009] show that applying the LPIPS loss on Cas-DM can effectively improve its performance, leading to the state-of-the-art image quality (measured by FID [Heusel *et al.*, 2017] and sFID [Nash *et al.*, 2021] on most datasets. Through a side-by-side visual comparison of Cas-DM with/without LPIPS using a fixed seed, we also discover that training diffusion models with the LPIPS loss makes the generated images have fewer artifacts as shown in Fig. 2. This work demonstrates that with a careful architecture design, metric functions such as the LPIPS loss can be used to improve the performance of diffusion models.

Our contributions are three-fold:

- We explore the methodology and efficacy of introducing extra metric functions into DDPM, resulting in a framework that can effectively incorporate metric functions during diffusion training.
- We introduce Cas-DM that addresses the main challenge in adding metric functions to DDPM by jointly predicting the added noise and the original clean image in each diffusion training and denoising step.
- Experiment results show that Cas-DM with the LPIPS loss consistently outperforms the state of the art across various datasets with different sampling steps.

2 Related Work

Denoising Diffusion Probabilistic Models. Starting from DDPM introduced by Ho *et al.*, diffusion models [Ho *et al.*, 2020; Dhariwal and Nichol, 2021; Nichol and Dhariwal, 2021; Rombach *et al.*, 2022] have outperformed GANs [Goodfellow *et al.*, 2014; Karras *et al.*, 2017; Mao *et al.*, 2017; Brock *et al.*, 2018; Wu *et al.*, 2019; Karras *et al.*, 2019; Karras *et al.*, 2020], Variational Auto-Encoders (VAE) [Kingma and Welling, 2013; Van Den Oord *et al.*, 2017; Vahdat and Kautz, 2020], auto-regressive models [Van Den Oord *et al.*, 2016b; Van den Oord *et al.*, 2016a; Salimans *et al.*, 2017; Chen *et al.*, 2018; Razavi *et al.*, 2019; Esser *et al.*, 2021], and normalization flows [Dinh *et al.*, 2014; Dinh *et al.*, 2017; Kingma and Dhariwal, 2018; Ho *et al.*, 2019] in terms of image quality while having a pretty stable training process. The diffusion model is in line with the score-based [Song and Ermon, 2019; Song and Ermon, 2020] and Markov-chains-based [Bengio *et al.*, 2014;

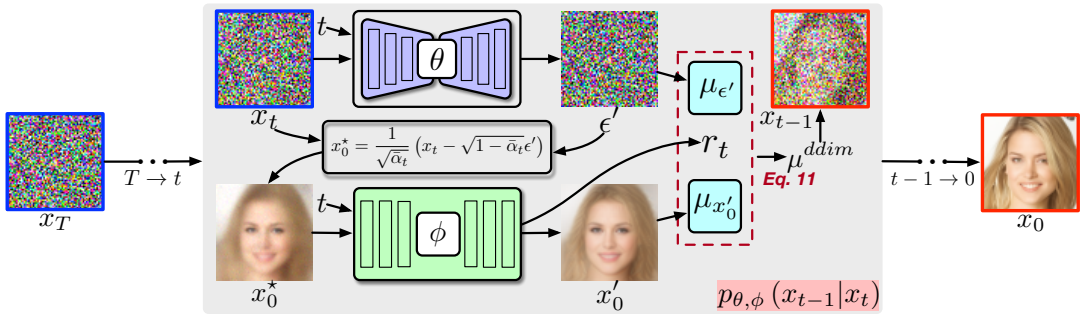


Figure 3: Framework of the proposed Cas-DM. For each time step t from T to 1, θ takes x_t and t as the inputs and estimates the added noise ϵ', which is then converted into an estimation of the clean image $x_0^\star$. Next, ϕ outputs the x_0' and r_t based on $x_0^\star$ and t, where the former is the final clean image estimation. r_t is then used to mix the μ estimations from x_0' and ϵ'. Cas-DM uses DDIM to run one backward step based on μ^{ddim}, getting x_{t-1}. Cas-DM runs the above process for $T-1$ rounds and gradually generates a clean image starting from a noise sample.

Salimans *et al.*, 2015] generative models, where the diffusion process can also be theoretically modeled by the discretization of a continuous SDE [Song *et al.*, 2020b]. Diffusion models have been used to generate multimedia content such as audio [Oord *et al.*, 2016], image [Brock *et al.*, 2018; Saharia *et al.*, 2022b; Ramesh *et al.*, 2022], and video [Singer *et al.*, 2022; Zhou *et al.*, 2022; Ho *et al.*, 2022; An *et al.*, 2023; Blattmann *et al.*, 2023]. The open-sourced latent diffusion model [Rombach *et al.*, 2022] sparks numerous image generation models based on conditions such as text [Saharia *et al.*, 2022b; Ramesh *et al.*, 2022; Yang *et al.*, 2023], sketch/segmentation maps [Rombach *et al.*, 2022; Fan *et al.*, 2023], and images in distinct domains [Saharia *et al.*, 2022a].

Improving Diffusion Models. The success of the diffusion model has drawn increasing interest in improving its algorithmic design. [Nichol and Dhariwal, 2021] improve the log-likelihood estimation and the generation quality of the DDPM by introducing a cosine-based noise schedule and letting the model learn variances of the reverse diffusion process in addition to the mean value in training. [Rombach *et al.*, 2022] introduce the latent diffusion model (LDM), which deploys the diffusion model on the latent space of an auto-encoder to reduce the computation cost. [Song *et al.*, 2020a] improve the sampling speed of the diffusion model by proposing an implicit diffusion model called DDIM. In terms of the architecture design, Benny and Wolf propose Dual Diffusion Model, which learns to predict ϵ and x_0 simultaneously in training, leading to improved generation quality. For the training approach, [Jolicoeur-Martineau *et al.*, 2020] explore adopting the adversarial loss as an extra loss to improve the prediction of x_0. This work studies an orthogonal improvement aspect of diffusion models – How to use additional metric functions to improve the generation performance. We draw the inspiration from [Jolicoeur-Martineau *et al.*, 2020] and [Benny and Wolf, 2022]. While [Jolicoeur-Martineau *et al.*, 2020] found that the adversarial objective based on a learnable discriminator is unnecessary for powerful generative models, we found that metric functions based on a fixed pre-trained network can achieve improved performance with a proper network architecture and training approach. The proposed diffusion model backbone shares the same idea of the dual output as [Benny and Wolf, 2022] but has different architectures and training strategies.

3 Preliminary

This section introduces the forward/backward diffusion processes and the training losses of DDPM, which will be used to derive the proposed Cas-DM later. The theory of diffusion models consists of a forward and a backward process. Given a clean image x_0 and a constant T to denote the maximum steps, the forward process gradually adds randomly sampled noise from a pre-defined distribution to x_0, leading to a sequence of images x_t for time steps $t \in [1, ..., T]$, where x_t is derived by adding noise to x_{t-1}. DDPM [Ho *et al.*, 2020] uses the Gaussian noise, resulting in the following transition equation,

$$q(x_t|x_{t-1}) := \mathcal{N}\left(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t \mathbf{I}\right), \quad (1)$$

where $\beta_t \in (0, 1]$ are pre-defined constants. Eq. 1 can derive a direct transition from x_0 to x_t,

$$q(x_t|x_0) := \mathcal{N}\left(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{I}\right), \quad (2)$$

where $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{i=1}^{t} \alpha_i$. Via Eq. 2, for any $t \in [1, T]$, one can easily get x_t given x_0 and a noise sample $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$,

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon. \quad (3)$$

Given a fixed x_t, Eq. 3 bridges x_0 and ϵ.

The backward process gradually recovers x_0 from the noisy image $x_T \in \mathcal{N}(x_T; \mathbf{0}, \mathbf{I})$, where for each t, the transition from x_t to x_{t-1} is $p(x_{t-1}|x_t)$, which is the ultimate target to learn of the diffusion model. The backward transition p is then approximated by

$$p_\theta(x_{t-1}|x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)). \quad (4)$$

The training objective is to maximize the variational lower bound (VLB) of the data likelihood. DDPM simplifies the training process to be first uniformly sample a t from $[1, T]$ and then compute,

$$L_t := D_{\mathrm{KL}}(q(x_{t-1}|x_t, x_0) \| p_\theta(x_{t-1}|x_t)), \quad (5)$$

which is further simplified to be

$$L_t := \frac{1}{2\beta_t^2} \|\tilde{\mu}_t(x_t, x_0, t) - \mu_\theta(x_t, t)\|^2, \quad (6)$$

where

$$\tilde{\mu}_t\left(x_t, x_0, t\right) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0 + \frac{\sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)}{1-\bar{\alpha}_t}x_t. \quad (7)$$

$\tilde{\mu}_t\left(x_t, x_0, t\right)$ and $\mu_\theta\left(x_t, t\right)$ are the mean values of $q\left(x_{t-1}|x_t, x_0\right)$ and $p_\theta\left(x_{t-1}|x_t\right)$, respectively. DDPM parameterizes μ_θ with a neural network θ that either predicts x_0 or ϵ, where two types of network outputs are denoted as x_0' and ϵ', respectively. We can obtain μ_θ via Eq. 7 with x_0'. If the network predicts ϵ', we first get an indirect x_0 prediction from ϵ' via Eq. 3. Then we can compute μ_θ via Eq. 7 with the indirect x_0 prediction. Ho *et al.* empirically demonstrate that ϵ' usually yields better image quality, i.e., lower FID score [Heusel *et al.*, 2017] than x_0'. One may refer to [Benny and Wolf, 2022] and [Ho *et al.*, 2020] for more detailed mathematical derivation.

4 Method

This section introduces the network architecture of our Cas-DM as well as its training and sampling processes.

4.1 Cascaded Diffusion Model

As shown in Fig. 3, the backbone of Cas-DM consists of two cascaded networks, denoted as θ and ϕ. θ is used to predict the added noise ϵ, and ϕ is to predict the clean image x_0. The architectures of both θ follow the improved diffusion [Nichol and Dhariwal, 2021] while ϕ is a network whose input and output tensors have the same shape. We use the model output from both θ and ϕ to obtain an estimation of $\mu_{\theta,\phi}\left(x_t, t\right)$, detailed as follows.

In training, θ takes the noisy image x_t and the uniformly sampled time step t as the input and predict the added noise ϵ', θ is equivalent to a vanilla DDPM predicting ϵ'. Based on Eq. 3, ϵ' can lead to an indirect estimation of x_0 as follows,

$$x_0^\star = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \sqrt{1-\bar{\alpha}_t}\epsilon'\right). \quad (8)$$

Next, based on ϵ', we obtain an estimation of $\mu_\theta\left(x_t, t\right)$, which is denoted as $\mu_{\epsilon'}\left(x_t, t\right)$,

$$\mu_{\epsilon'}\left(x_t, t\right) := \frac{1}{\sqrt{\alpha_t}}x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\epsilon'. \quad (9)$$

Eq. 9 is derived by replacing x_0 in Eq. 7 with $x_0^\star$ in Eq. 8.

ϕ takes $x_0^\star$ and t as the input and output x_0' as well as a dynamic value r_t, which is used to balance the strength of θ and ϕ in computing $\mu_{\theta,\phi}\left(x_t, t\right)$ later. The application of r_t is directly inspired by dual diffusion [Benny and Wolf, 2022], where their experiments show that r_t can better balance the effects of two types of predictions and lead to improved performance. We follow this setting and obtain r_t by adding an extra channel to the output layer of ϕ. The output of ϕ is the concatenation of $x_0' \in \mathbb{R}^{H,W,C}$ and $r_t \in \mathbb{R}^{H,W,1}$ along the channel dimension. We obtain the estimation of $\mu_\theta\left(x_t, t\right)$ based on x_0' as

$$\mu_{x_0'}\left(x_0^\star, t\right) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0' + \frac{\sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)}{1-\bar{\alpha}_t}x_t. \quad (10)$$

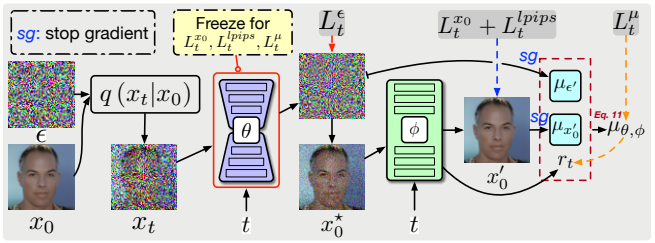


Figure 4: Training process of Cas-DM. θ learns to estimate the added noise ϵ while ϕ is trained to predict the clean image x_0. We apply L_t^ϵ on θ and all the gradients of other losses are blocked for it. For ϕ, we use $L_t^{x_0}$, L_t^{lpips}, and L_t^μ losses, where the first two is to enforce ϕ to recover the clean image from $x_0^\star$, assisted by the the LPIPS loss. L_t^μ is to train the dynamic mixing weight and the gradient is stopped before $\mu_{\epsilon'}$ and $\mu_{x_0'}$. Best viewed on screen by zoom-in.

ϕ is to improve the accuracy of the x_0 prediction on top of θ's output. The final estimation of $\mu_{\theta,\phi}\left(x_t, t\right)$ is

$$\mu_{\theta,\phi}\left(x_t, t\right) = r_t \cdot \mu_{x_0'}\left(x_t, t\right) + \left(1-r_t\right) \cdot \mu_{\epsilon'}\left(x_t, t\right) \quad (11)$$

Compared with dual diffusion [Benny and Wolf, 2022], Cas-DM allows the dedicated metric functions to be applied on the x_0 branch without influencing the ϵ branch because we could stop the gradients of metric functions on θ. More details will be in the next part.

4.2 Training and Sampling

As shown in Fig. 4, we train Cas-DM following the approach of dual diffusion [Benny and Wolf, 2022] with the following loss terms,

$$\begin{aligned} L_t^\epsilon &= \left\|\epsilon - \epsilon'\right\|^2, \\ L_t^{x_0} &= \left\|x_0 - x_0'\right\|^2, \\ L_t^\mu &= \left\|\tilde{\mu}_t - \left(r_t\left[\mu_{x_0'}\right]_{\text{sg}} + \left(1-r_t\right)\left[\mu_{\epsilon'}\right]_{\text{sg}}\right)\right\|^2. \end{aligned} \quad (12)$$

The input value of μ_t, $\mu_{x_0'}$, and $\mu_{\epsilon'}$ are omitted for simplicity. $\left[\cdot\right]_{\text{sg}}$ denotes stop gradients. We use the LPIPS loss [Johnson *et al.*, 2016] from the `piq` repository* in training to demonstrate that extra metric functions can be applied to Cas-DM for further improvements,

$$L_t^{lpips} = \text{LPIPS}\left(\mathcal{T}\left(x_0\right), \mathcal{T}\left(x_0'\right)\right). \quad (13)$$

Here $\mathcal{T}$ denotes an image transformation module, which first interpolates an image to the size of 224×224 with the bilinear interpolation, then linearly normalize its value to the range of $[0, 1]$. In back-propagating, we disconnect θ and ϕ by detaching the whole θ from the computing graph of ϕ, leading to separate loss functions for θ and ϕ:

$$L_t^\theta = \lambda^\epsilon L_t^\epsilon, \quad (14)$$

$$L_t^\phi = \lambda^{x_0} L_t^{x_0} + \lambda^\mu L_t^\mu + \lambda^{lpips} L_t^{lpips}. \quad (15)$$

Since the LPIPS loss only works well on real images, we use L_t^θ to let θ learn to predict ϵ without the disturbance of the

*https://github.com/photosynthesis-team/piq

Model	FID↓	sFID↓
CIFAR10 32×32		
⋆Gated PixelCNN [Van den Oord *et al.*, 2016a]	65.93	-
⋆EBM [Song and Kingma, 2021]	38.20	-
⋆NCSNv2 [Song and Ermon, 2020]	31.75	-
⋆SNGAN-DDLS [Che *et al.*, 2020]	15.42	-
⋆StyleGAN2 + ADA (v1) [Karras *et al.*, 2020]	3.26	-
⋆DDPM [Ho *et al.*, 2020]	32.65	-
⋆DDIM [Song *et al.*, 2020a]	5.57	-
⋆Improved DDPM [Nichol and Dhariwal, 2021]	4.58	-
⋆Improved DDIM [Nichol and Dhariwal, 2021]	6.29	-
⋆Dual Diffusion [Benny and Wolf, 2022]	5.10	-
⋆Consistency Model (CD) [Benny and Wolf, 2022]	2.93	-
⋆Consistency Model (CT) [Benny and Wolf, 2022]	5.83	-
DDPM (ϵ mode)	6.79	4.97
DDPM (x_0 mode)	17.78	6.69
DDPM (x_0 + LPIPS)	9.34 (-8.44)	6.94 (+0.25)
Dual Diffusion	6.52	4.60
Dual Diffusion + LPIPS	**5.65** (-0.87)	4.89 (+0.29)
Cas-DM [$\phi_{\texttt{UNet}}$]	6.80	5.03
Cas-DM [$\phi_{\texttt{UNet}}$] + LPIPS	6.40 (-0.40)	4.87 (-0.16)
Cas-DM [$\phi_{\texttt{Fix-Res}}$]	6.40	4.60
Cas-DM [$\phi_{\texttt{Fix-Res}}$] + LPIPS	6.28 (-0.12)	**4.57** (-0.03)
LSUN Bedroom 64×64		
DDPM (ϵ mode)	5.51	27.61
DDPM (x_0 mode)	10.28	32.13
DDPM (x_0 + LPIPS)	13.14 (+2.86)	32.93 (+0.80)
Dual Diffusion	5.49	27.71
Dual Diffusion + LPIPS	7.72 (+2.23)	29.08 (+1.37)
Cas-DM [$\phi_{\texttt{UNet}}$]	5.29	27.80
Cas-DM [$\phi_{\texttt{UNet}}$] + LPIPS	**5.17** (-0.12)	**27.45** (-0.35)

Model	FID↓	sFID↓
CelebAHQ 64×64		
⋆DDPM [Ho *et al.*, 2020]	43.90	-
⋆DDIM [Song and Ermon, 2020]	6.15	-
⋆Dual Diffusion [Benny and Wolf, 2022]	4.07	-
DDPM (ϵ mode)	6.34	17.16
DDPM (x_0 mode)	8.82	19.11
DDPM (x_0 + LPIPS)	10.54 (+1.72)	21.00 (+1.89)
Dual Diffusion	5.47	15.26
Dual Diffusion + LPIPS	6.86 (+1.39)	16.06 (+0.80)
Cas-DM [$\phi_{\texttt{UNet}}$]	5.33	14.87
Cas-DM [$\phi_{\texttt{UNet}}$] + LPIPS	4.95 (-0.38)	14.71 (-0.16)
Cas-DM [$\phi_{\texttt{Fix-Res}}$]	5.07	14.77
Cas-DM [$\phi_{\texttt{Fix-Res}}$] + LPIPS	**4.64** (-0.43)	**14.32** (-0.45)
ImageNet 64×64		
with guidance		
⋆BigGAN-deep [Brock *et al.*, 2018]	4.06	3.96
⋆Improved DDPM [Nichol and Dhariwal, 2021]	2.92	3.79
⋆ADM [Dhariwal and Nichol, 2021]	2.61	3.77
⋆ADM (dropout) [Dhariwal and Nichol, 2021]	2.07	4.29
without guidance		
⋆Consistency Model (CD) [Benny and Wolf, 2022]	4.70	-
⋆Consistency Model (CT) [Benny and Wolf, 2022]	11.10	-
DDPM (ϵ mode)	27.96	18.73
DDPM (x_0 mode)	65.09	23.86
DDPM (x_0 + LPIPS)	41.41 (-23.68)	28.20 (+4.34)
Dual Diffusion	38.65	18.38
Dual Diffusion + LPIPS	31.84 (-6.81)	21.88 (+3.50)
Cas-DM [$\phi_{\texttt{UNet}}$]	28.34	18.46
Cas-DM [$\phi_{\texttt{UNet}}$] + LPIPS	**27.54** (-0.80)	**18.06** (-0.40)

Table 1: Performance comparison of Cas-DM variants and the baseline models on CIFAR10, CelebAHQ, LSUN Bedroom, and ImageNet. The best and second best results are marked with **bold** and underline, respectively. Cas-DM [$\phi_{\texttt{UNet}}$] and Cas-DM [$\phi_{\texttt{Fix-Res}}$] denote using UNet and a fixed-resolution CNN as the backbone of the ϕ module in Cas-DM, respectively. Models marked with ⋆ are borrowed from [Ho *et al.*, 2020], [Benny and Wolf, 2022], [Dhariwal and Nichol, 2021], and [Song *et al.*, 2023], which are **for reference and not directly comparable** with other models due to the different diffusion model implementation, training and sampling settings, dataset preparation, and FID evaluation settings.

gradient from ϕ and the metric functions, leading to a stable μ_θ estimation, $\mu_{\epsilon'}$, as the basis. On top of it, ϕ learns to predict x_0, resulting in another estimation $\mu_{\epsilon'}$. The LPIPS loss can improve the accuracy of $\mu_{\epsilon'}$, leading to an overall better μ_θ estimation through Eq. 11.

We use DDIM [Song *et al.*, 2020a] for sampling. Following dual diffusion [Benny and Wolf, 2022], we obtain the μ estimation via the DDIM's μ computing equation from ϵ' and x'_0, respectively,

$$\mu^{ddim}_{x'_0} = \sqrt{\bar{\alpha}_{t-1}}x'_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\cdot\frac{x_t-\sqrt{\bar{\alpha}_t}x'_0}{\sqrt{1-\bar{\alpha}_t}}, \quad (16)$$

$$\mu^{ddim}_{\epsilon'} = \frac{x_t-\sqrt{1-\bar{\alpha}_t}\epsilon'}{\sqrt{\alpha_t}} + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\cdot\epsilon'. \quad (17)$$

Then the final estimation of the μ^{ddim} for DDIM sampling is the interpolation of $\mu^{ddim}_{x'_0}$ and $\mu^{ddim}_{\epsilon'}$ based on r_t.

5 Experiments

5.1 Implementation Details

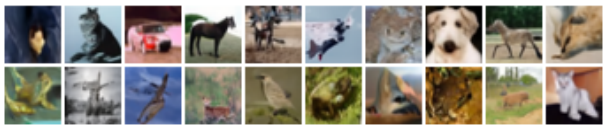
Diffusion Model. We implement Cas-DM based on the official code† of improved diffusion [Nichol and Dhariwal, 2021]. θ is the default U-Net architecture with 128 channels, 3 ResNet blocks per layer, and the learn sigma flag disabled. For hyper-parameters of diffusion models, we use 4000 diffusion steps with the cosine noise scheduler in all experiments, where the KL loss is not used.

†https://github.com/openai/improved-diffusion

Model	CIFAR10	CelebAHQ	LSUN Bedroom	ImageNet
DDPM (x_0 mode)	17.78	8.82	10.28	65.09
+ LPIPS	-8.44	+1.72	+2.86	-24.68
Dual Diffusion	6.52	5.47	5.49	38.65
+ LPIPS	-0.87	+1.39	+2.23	-6.81
Cas-DM [$\phi_{\texttt{UNet}}$]	6.80	5.33	5.29	28.34
+ LPIPS	-0.40	-0.38	-0.12	-0.80

Table 2: FID variation comparison after applying the LPIPS loss. The FID values of DDPM and Dual Diffusion Model fluctuate after applying the LPIPS loss. Cas-DM [$\phi_{\texttt{UNet}}$] achieves consistent improvement across all the compared datasets.

Metric Function. We use the LPIPS loss as a prototype metric function following consistency model [Song *et al.*, 2023], where we replace all MaxPooling layers of the LPIPS backbone with AveragePooling operations.



(a) CIFAR10 samples. FID=6.40

(b) CelebAHQ samples. FID=4.95

(c) LSUN Bedroom samples. FID=5.17

(d) ImageNet samples. FID=27.54

Figure 5: Unconditional samples from Cas-DM [$\phi_{\texttt{UNet}}$] trained with the LPIPS loss on the experimented datasets.

Model	FID↓	sFID↓
CelebAHQ 64×64		
Dual Diffusion*	11.41	19.91
Dual Diffusion* + LPIPS	8.49 (-2.92)	20.10 (+0.19)
Dual Diffusion†	5.90	15.71
Dual Diffusion† + LPIPS	7.62 (+1.71)	16.55 (+0.84)
Cas-DM [$\phi_{\texttt{UNet}}$]	5.33	14.87
Cas-DM [$\phi_{\texttt{UNet}}$] + LPIPS	**4.95** (-0.38)	**14.71** (-0.16)
LSUN Bedroom 64×64		
Dual Diffusion*	9.71	31.64
Dual Diffusion* + LPIPS	13.80 (+4.09)	34.46 (+2.82)
Dual Diffusion†	6.79	28.84
Dual Diffusion† + LPIPS	9.64 (+2.85)	30.23 (+1.39)
Cas-DM [$\phi_{\texttt{UNet}}$]	6.63	29.11
Cas-DM [$\phi_{\texttt{UNet}}$] + LPIPS	**6.34** (-0.29)	**28.71** (-0.40)

Table 3: Performance Comparison of Cas-DM [$\phi_{\texttt{UNet}}$] and the variants of Dual Diffusion Models. All the reported scores are based on the best checkpoints of 100k iterations. * denotes doubling the channel of all UNet layers and † represents cascading two UNets.

Training. Training is conducted on 8 V100 GPUs with 32GB GPU RAM, where the batch size for each GPU is 16, leading to 128 accumulated batch size. We set learning rate to $1e^{-4}$ with no learning rate decay. When computing loss functions, λ^{ϵ}, λ^{x_0}, and λ^{μ} are set to 1.0 while λ^{lpips} is set to 0.1. We train the model for 400k iterations and perform sampling and evaluation with the gap of 20k and 100k when the iteration is less than and higher than 100k, respectively. For each model, we report the best result among all the evaluated checkpoints.

Sampling. When sampling, we use the DDIM sampler and re-space the diffusion step to 100. For each checkpoint, we sample 50k images for CIFAR10 and 10k images for other datasets and compute the evaluation metrics with respect to the training dataset.

5.2 Experiment Settings

Datasets. We conduct experiments on the CIFAR10, CelebAHQ, LSUN Bedroom, and ImageNet datasets. We train the model with image size 32×32 on CIFAR10 and 64×64 on the others.

Metrics. We compare models with Fréchet Inception Distance (FID) and sFID. FID and sFID evaluate the distributional similarity between the generated and training images. FID is based on the `pool_3` feature of Inception V3 [Szegedy *et al.*, 2016], while sFID uses `mixed_6/conv` feature maps. sFID is more sensitive to spatial variability [Nash *et al.*, 2021].

Baselines. We compare Cas-DM with a few baselines.

- **DDPM (ϵ mode).** We train DDPM by letting the U-Net predict the added noise ϵ, which is then used to generate images with the DDIM sampler.
- **DDPM (x_0 mode).** It is similar to DDPM (ϵ mode), where the U-Net predicts the clean image x_0.
- **DDPM (x_0) + LPIPS.** We add the LPIPS loss in the training of the DDPM (x_0 mode). This model is to verify whether adding metric functions can improve the performance of DDPM.
- **Dual Diffusion.** We re-implement the Dual Diffusion Model based on the official code of the improved diffusion and then train the model with the same setting as other baselines.
- **Dual Diffusion + LPIPS.** We train the re-implemented Dual Diffusion Model by adding the LPIPS loss to its x_0 prediction. This model is to verify whether adding metric functions can improve the performance of the Dual Diffusion Model.

We compare the proposed Cas-DM with the above baselines by conducting two experiments.

- **Cas-DM [$\phi_{\texttt{UNet}}$] & Cas-DM [$\phi_{\texttt{Fix-Res}}$].** We train Cas-DM with the same settings as other baselines. We consider two variants of Cas-DM, which use UNet and a fixed-resolution CNN as ϕ's backbones, respectively. This experiment is to demonstrate the performance of the vanilla Cas-DM without adding any metric function.
- **Cas-DM [$\phi_{\texttt{UNet}}$] & Cas-DM [$\phi_{\texttt{Fix-Res}}$] + LPIPS.** We add the LPIPS loss to the x_0' head of Cas-DM to verify whether the new diffusion model architecture enables the successful application of the LPIPS loss.

5.3 Main Results

Qualitative Comparison. We conducted a one-by-one visual comparison of Cas-DM with/without LPIPS on CelebAHQ with fixed seed. As shown in Fig. 2, using metric functions (LPIPS) in diffusion model training makes the generated images have fewer artifacts. For example, in the first

Model	CIFAR10	CelebAHQ
Cas-DM [$\phi_{\texttt{UNet}}$]	6.80	5.33
Cas-DM [$\phi_{\texttt{UNet}}$] + LPIPS (VGG)	**6.40**	**4.95**
Cas-DM [$\phi_{\texttt{UNet}}$] + ResNet	6.64	5.67
Cas-DM [$\phi_{\texttt{UNet}}$] + Inception	6.94	5.52
Cas-DM [$\phi_{\texttt{UNet}}$] + Swin	6.98	5.55

Table 4: FID comparison between pre-trained backbones of the metric functions.

Model	CIFAR10	CelebAHQ
Cas-DM [$\phi_{\texttt{Fix-Res}}$] Input: $x_0^\star$	**6.40**	**5.07**
Cas-DM [$\phi_{\texttt{Fix-Res}}$] Input: $\texttt{cancat}\,(x_0^\star, \epsilon')$	7.08	5.78

Table 5: FID comparison between ϕ's input settings.

and fourth columns, using LPIPS corrected the artifacts in generation eye glasses (col 1/4), hair (col 2), and face (col 3). Fig. 5 shows more results generated by Cas-DM [$\phi_{\texttt{UNet}}$] with LPIPS (the same model in Table 1) using random seed.

Quantitative Comparison. We compare the unconditional image generation performance of Cas-DM [$\phi_{\texttt{UNet}}$] with baselines in Table 1. We additional list the results in existing papers for reference, which are not comparable since they use different training and sampling settings. For CelebAHQ, LSUN Bedroom, and ImageNet, Cas-DM + LPIPS achieves the best FID and sFID scores among all the compared methods. This indicates that the architecture of Cas-DM is valuable compared with DDPM and Dual Diffusion Model since it produces better results than others on many datasets. More importantly, the improved performance achieves by Cas-DM (both two variants on ϕ backbone) + LPIPS indicates that adding metric functions such as LPIPS is a meaningful strategy to improve the performance of diffusion models, where Cas-DM shows a feasible diffusion model architecture design that can make it work.

Metric Function Effectiveness. Table 2 compares the performance of diffusion models with and without the LPIPS loss. We list the FID increase or drop after the usage of the LPIPS loss on DDPM in x_0 mode, Dual Diffusion Model, and Cas-DM [$\phi_{\texttt{UNet}}$]. DDPM (x_0 model) and Dual Diffusion Model achieve improved performance (reduced FID score) after applying the LPIPS loss on CIFAR10 and ImageNet. However, the results are inconsistent on the other two datasets. The proposed Cas-DM [$\phi_{\texttt{UNet}}$] achieves consistently improved performance among all the compared datasets. This indicates that the architectural design of Cas-DM enables the effective application of the LPIPS loss on diffusion model training.

Cas-DM *v.s.* Dual Diffusion Variants. To demonstrate that the better performance of Cas-DM against Dual Diffusion Models [Benny and Wolf, 2022] comes from the novel architecture design rather than more trainable parameters, we conducted experiments by scaling up Dual Diffusion Models to the same parameter size as our Cas-DM and comparing their performance on CelebAHQ and LSUN Bedroom datasets. We consider two scaling-up approaches: 1) Doubling the channel of all UNet layers (marked with *) and 2) cascading two UNets (marked with †) as Cas-DM does. As shown in Tab. 3, either enlarging channels or using two UNets cannot improve the performance of Dual Diffusion Models. Moreover, the LPIPS metric function does not work on most Dual Diffusion Model variants, demonstrating the Cas-DM's effectiveness in enabling metric functions. More results on other datasets will be added to the revised paper.

Model	CIFAR10		CelebAHQ	
	Step 10	Step 100	Step 10	Step 100
DDPM (ϵ mode)	16.57	6.79	27.76	6.34
Cas-DM [$\phi_{\texttt{UNet}}$]	14.91	6.80	28.67	5.32
Cas-DM [$\phi_{\texttt{UNet}}$] + LPIPS	**13.75** (-1.16)	**6.40** (-0.40)	**27.36** (-1.31)	**4.95** (-0.37)

Table 6: FID comparison between different sampling steps.

5.4 Ablation Study

We conduct an ablation study on metric function's backbones, the input settings of network ϕ in Cas-DM, and the DDIM sampling steps.

Metric Function Backbones. Our experimental results have demonstrated that the LPIPS loss can improve the performance of diffusion models. Table 4 compares the LPIPS loss with other metric function backbones. We use the ResNet [He *et al.*, 2016], Inception v3 [Szegedy *et al.*, 2016], and Swin Transformer [Liu *et al.*, 2021] pre-trained on the ImageNet dataset. Similar to the LPIPS loss, we first extract their features on images at different layers, which is then used to compute the mean square error between corresponding feature maps. We find that ResNet can improve the performance on CIFAR10 while others do not work. We conjecture that the VGG network [Simonyan and Zisserman, 2014] used by the LPIPS loss does not use residual connections, which may make the extracted features contain more semantic information. Similar observations have also been made by [Karras *et al.*, 2020]. We leave the discovery of more powerful metric functions and other strategies for improving the diffusion model training as future work.

Input Types of ϕ. With the motivation that ϵ' can influence the appearance of $x_0^\star$ according to Eq. 8, we also attempt to take the concatenation of $x_0^\star$ and ϵ' as the input to train ϕ, which we find does not work well in terms of FID.

Sampling Steps. Table 6 compares the FID of DDPM in the ϵ mode and Cas-DM [$\phi_{\texttt{UNet}}$] with/without the LPIPS loss on sampling steps 10 and 100. Cas-DM works equally well on small and large sampling steps in terms of enabling the successful application of the LPIPS loss. Note that on different datasets, a comparably larger performance gain may be achieved by either smaller or larger sampling steps.

6 Conclusion

In this paper, we study using metric functions to improve the performance of image diffusion models. To this end, we propose Cas-DM, which uses two cascaded networks to predict the added noise and the clean image, respectively. This architecture design addresses the issue of the dual diffusion model where the LPIPS affects the noise prediction.Experimental results on several datasets show that Cas-DM assisted with the LPIPS loss achieves the state-of-the-art results.

References

[An *et al.*, 2023] Jie An, Songyang Zhang, Harry Yang, Sonal Gupta, Jia-Bin Huang, Jiebo Luo, and Xi Yin. Latent-shift: Latent diffusion with temporal shift for efficient text-to-video generation. *arXiv preprint arXiv:2304.08477*, 2023.

[Bengio *et al.*, 2014] Yoshua Bengio, Eric Laufer, Guillaume Alain, and Jason Yosinski. Deep generative stochastic networks trainable by backprop. In *International Conference on Machine Learning*, pages 226–234. PMLR, 2014.

[Benny and Wolf, 2022] Yaniv Benny and Lior Wolf. Dynamic dual-output diffusion models. In *CVPR*, 2022.

[Blattmann *et al.*, 2023] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *CVPR*, 2023.

[Brock *et al.*, 2018] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[Che *et al.*, 2020] Tong Che, Ruixiang Zhang, Jascha Sohl-Dickstein, Hugo Larochelle, Liam Paull, Yuan Cao, and Yoshua Bengio. Your gan is secretly an energy-based model and you should use discriminator driven latent sampling. *NeurIPS*, 2020.

[Chen *et al.*, 2018] Xi Chen, Nikhil Mishra, Mostafa Rohaninejad, and Pieter Abbeel. Pixelsnail: An improved autoregressive generative model. In *ICML*, 2018.

[Deng *et al.*, 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: a large-scale hierarchical image database. In *CVPR*, 2009.

[Dhariwal and Nichol, 2021] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In *NeurIPS*, 2021.

[Dinh *et al.*, 2014] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[Dinh *et al.*, 2017] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In *ICLR*, 2017.

[Esser *et al.*, 2021] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021.

[Fan *et al.*, 2023] Wan-Cyuan Fan, Yen-Chun Chen, Dong-Dong Chen, Yu Cheng, Lu Yuan, and Yu-Chiang Frank Wang. Frido: Feature pyramid diffusion for complex scene image synthesis. In *AAAI*, 2023.

[Goodfellow *et al.*, 2014] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[Heusel *et al.*, 2017] 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 *NeurIPS*, 2017.

[Ho *et al.*, 2019] Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *ICML*, 2019.

[Ho *et al.*, 2020] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 2020.

[Ho *et al.*, 2022] 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.

[Johnson *et al.*, 2016] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016.

[Jolicoeur-Martineau *et al.*, 2020] Alexia Jolicoeur-Martineau, Rémi Piché-Taillefer, Rémi Tachet des Combes, and Ioannis Mitliagkas. Adversarial score matching and improved sampling for image generation. *arXiv preprint arXiv:2009.05475*, 2020.

[Karras *et al.*, 2017] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[Karras *et al.*, 2019] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.

[Karras *et al.*, 2020] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *CVPR*, 2020.

[Kingma and Dhariwal, 2018] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *NeurIPS*, 2018.

[Kingma and Welling, 2013] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

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

[Liu *et al.*, 2021] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021.

[Mao *et al.*, 2017] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *ICCV*, 2017.

[Nash *et al.*, 2021] Charlie Nash, Jacob Menick, Sander Dieleman, and Peter W Battaglia. Generating images with sparse representations. *arXiv preprint arXiv:2103.03841*, 2021.

[Nichol and Dhariwal, 2021] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *ICML*, 2021.

[Oord *et al.*, 2016] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[Ramesh *et al.*, 2022] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[Razavi *et al.*, 2019] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *NeurIPS*, 2019.

[Rombach *et al.*, 2022] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

[Saharia *et al.*, 2022a] Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *SIGGRAPH*, 2022.

[Saharia *et al.*, 2022b] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *NeurIPS*, 2022.

[Salimans *et al.*, 2015] Tim Salimans, Diederik Kingma, and Max Welling. Markov chain monte carlo and variational inference: Bridging the gap. In *ICML*, 2015.

[Salimans *et al.*, 2017] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

[Simonyan and Zisserman, 2014] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[Singer *et al.*, 2022] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, et al. Make-a-video: Text-to-video generation without text-video data. *arXiv preprint arXiv:2209.14792*, 2022.

[Song and Ermon, 2019] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *arXiv preprint arXiv:1907.05600*, 2019.

[Song and Ermon, 2020] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *arXiv preprint arXiv:2006.09011*, 2020.

[Song and Kingma, 2021] Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.

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

[Song *et al.*, 2020b] 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*, 2020.

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

[Szegedy *et al.*, 2016] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

[Vahdat and Kautz, 2020] Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *NeurIPS*, 2020.

[Van den Oord *et al.*, 2016a] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. *NeurIPS*, 2016.

[Van Den Oord *et al.*, 2016b] Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *ICML*, 2016.

[Van Den Oord *et al.*, 2017] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *NeurIPS*, 2017.

[Wu *et al.*, 2019] Yan Wu, Jeff Donahue, David Balduzzi, Karen Simonyan, and Timothy Lillicrap. Logan: Latent optimisation for generative adversarial networks. *arXiv preprint arXiv:1912.00953*, 2019.

[Yang *et al.*, 2023] Zhengyuan Yang, Jianfeng Wang, Zhe Gan, Linjie Li, Kevin Lin, Chenfei Wu, Nan Duan, Zicheng Liu, Ce Liu, Michael Zeng, et al. Reco: Region-controlled text-to-image generation. In *CVPR*, 2023.

[Yu *et al.*, 2015] 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.

[Zhang *et al.*, 2018] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[Zhou *et al.*, 2022] Daquan Zhou, Weimin Wang, Hanshu Yan, Weiwei Lv, Yizhe Zhu, and Jiashi Feng. Magicvideo: Efficient video generation with latent diffusion models. *arXiv preprint arXiv:2211.11018*, 2022.