Diffusion Study Group #6

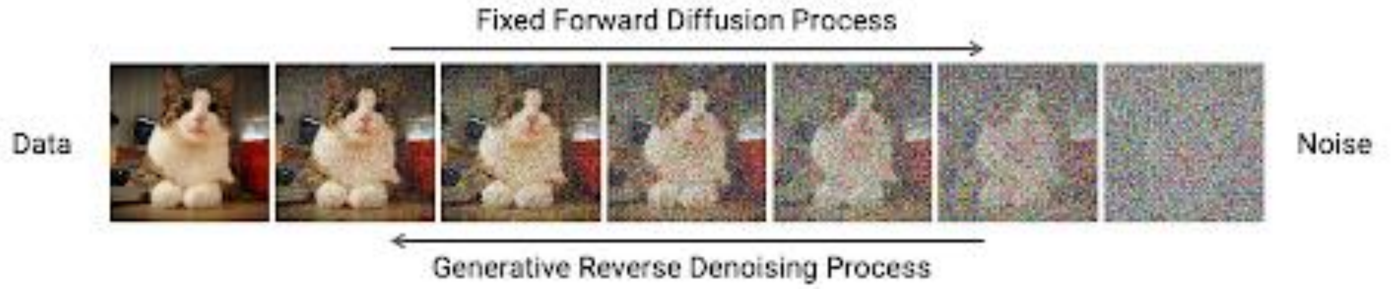
10/22/2022

Tanishq Abraham

Review and Q&A

What are diffusion models? – a short summary

- Originally invented in 2015 by Jascha Sohl-Dickstein et al.
- Gained prominence thanks to DDPMs in 2020 – first demo of high quality samples
- A short, simplified summary: We train a neural network to iteratively denoise a sample starting from pure noise, which can be shown to be equivalent to sampling from the estimated data distribution
- Still a very new field!

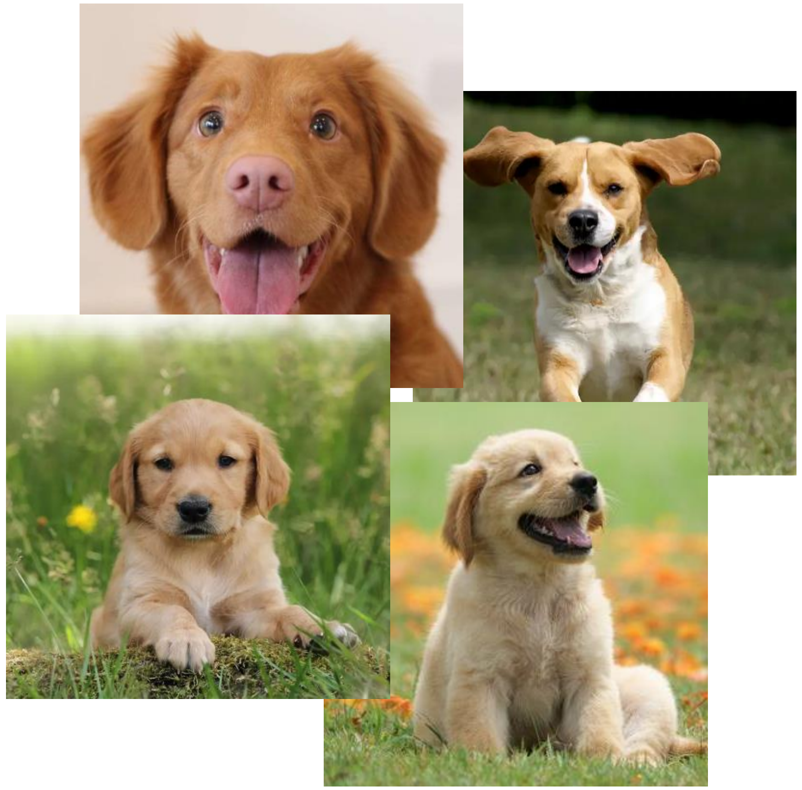

Fixed Forward Diffusion Process
Data
Noise
Generative Reverse Denoising Process

Denoising Diffusion Probabilistic Models

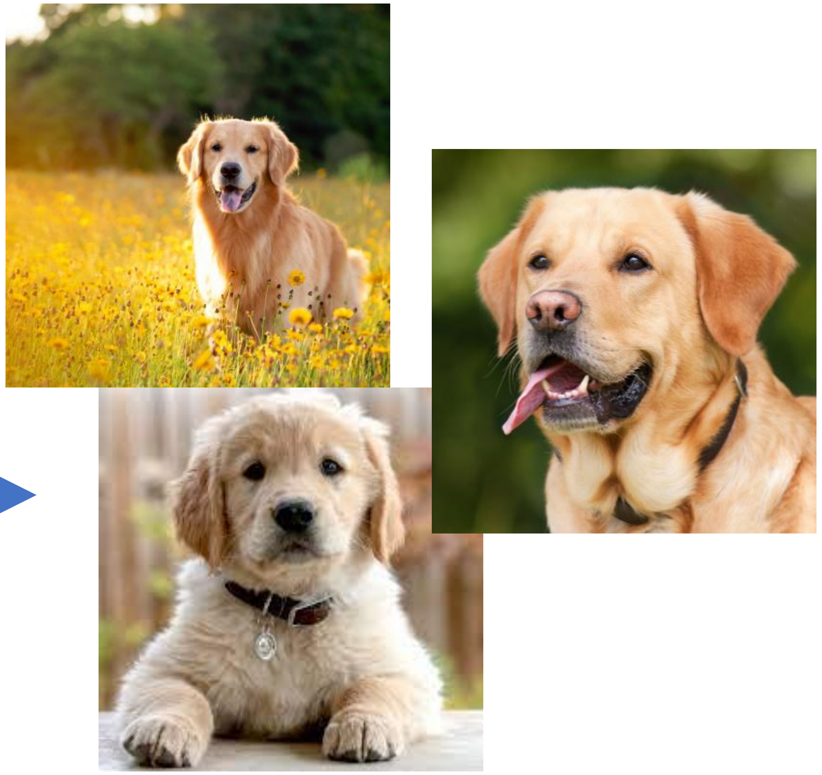
Jonathan Ho, Ajay Jain, Pieter Abbeel

June 19th, 2020

What’s the task?



Given these datapoints…

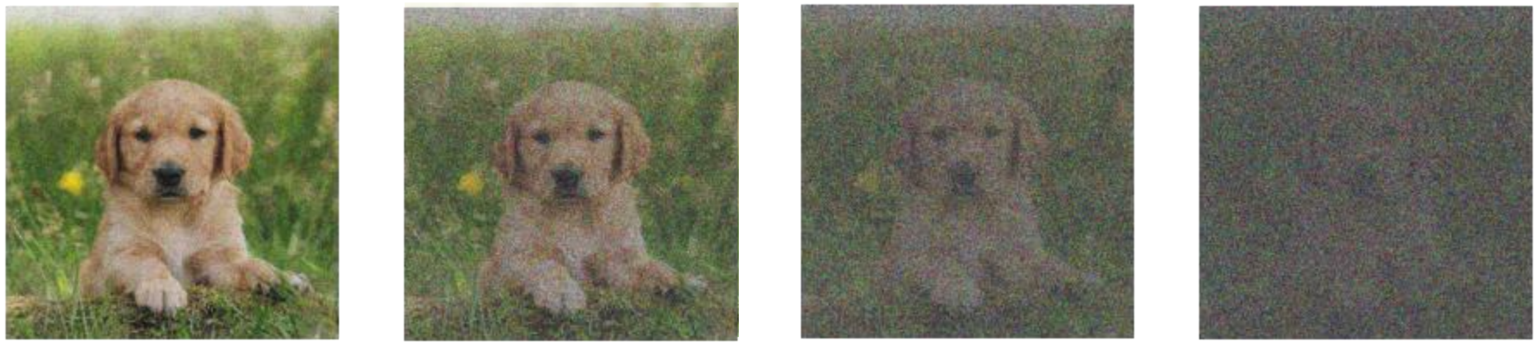


Can we generate more like it?

Forward process

$$q(\mathbf{x}_t|\mathbf{x}_0) := \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$



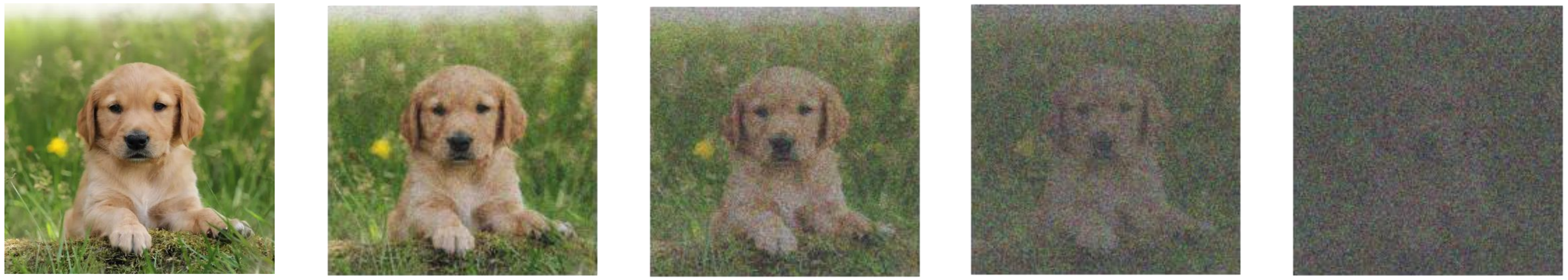
$\mathbf{x}_{25}$ $\mathbf{x}_{50}$ / $\mathbf{x}_0$ $\mathbf{x}_{75}$ $\mathbf{x}_{100} = \mathbf{x}_T$

Observed image

Equivalent to Gaussian noise

Reverse process

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}\left(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)\right)$$



$\mathbf{x}_0$ $\mathbf{x}_{25}$ $\mathbf{x}_{50}$ $\mathbf{x}_{75}$ $\mathbf{x}_{100} = \mathbf{x}_T$

Generated image!

Equivalent to Gaussian noise

How to train diffusion models?

Want to find the most optimal parameters of model that maximize likelihood of training data:

$$\theta^\star = \arg\max_{\theta\in\Theta} p_\theta(\mathbf{x}_{0:T})$$

Therefore we must minimize:

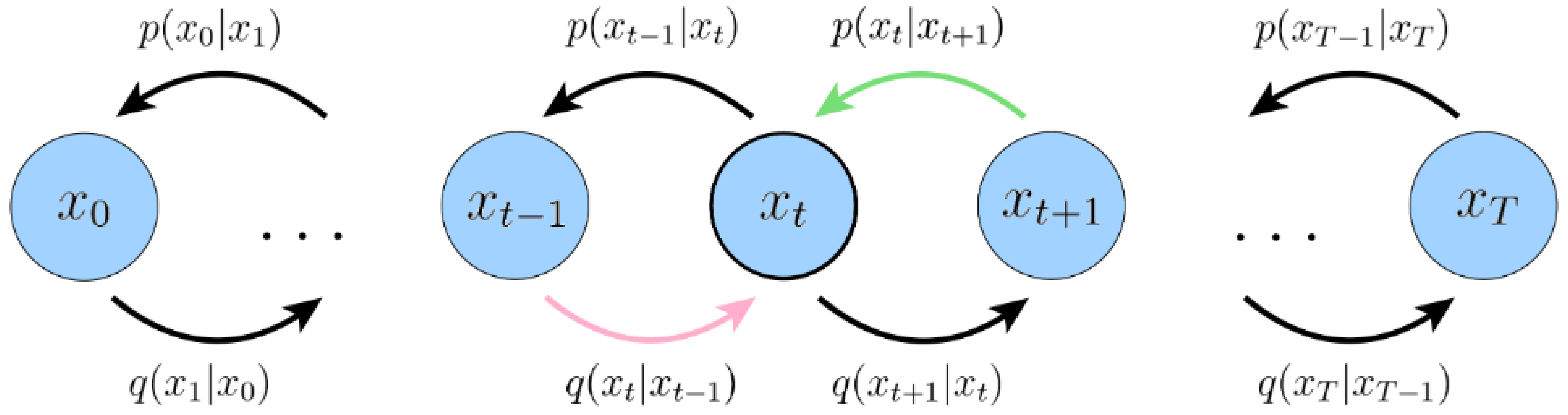
$$\mathcal{L} = E_{\mathbf{x}_0\sim p_{data}}[-\log p_\theta(\mathbf{x}_0)]$$

This requires:

$$p_\theta(\mathbf{x}_0) = \int p_\theta(\mathbf{x}_0|\mathbf{x}_{1:T})\mathrm{d}\mathbf{x}_{1:T}$$

which is intractable!

Latent variable perspective – Hierarchical variational autoencoder



A diffusion model can be considered as a hierarchical VAE with a fixed Gaussian encoder

Model objective

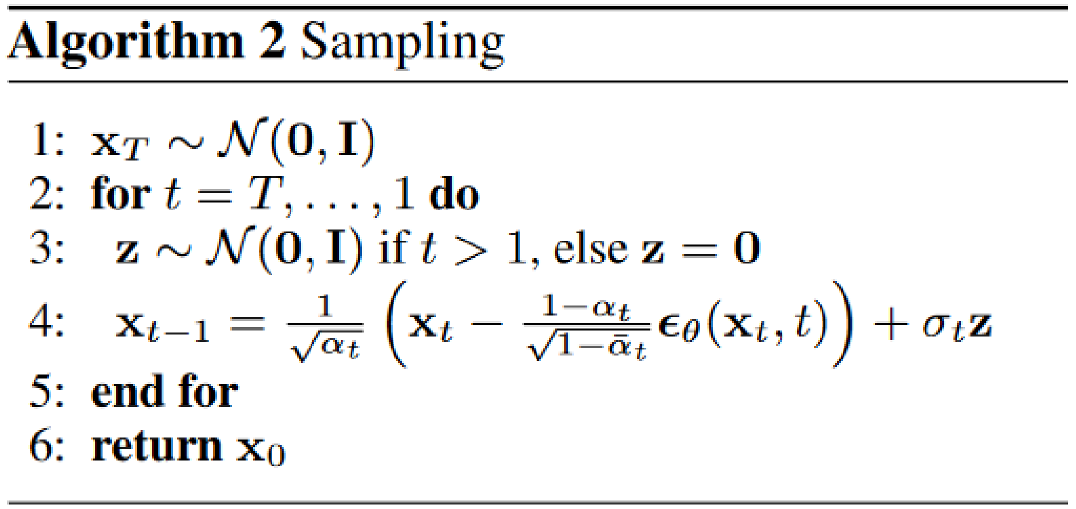
Denoising objective:

$$L(\theta) = E_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}[||\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)||^2]$$

where:

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

Model sampling



Algorithm 2 Sampling

```
1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: for $t = T, \ldots, 1$ do
3:   $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4:   $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: end for
6: return $\mathbf{x}_0$
```

Equivalent to a two-step process:

1. Given $\mathbf{x}_t, \boldsymbol{\epsilon}_{\boldsymbol{\theta}}, t$, get predicted denoised image $\mathbf{x}_0 \approx \hat{\mathbf{x}}_0 = \left(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t)\right) / \sqrt{\bar{\alpha}_t}$
2. Plug $\hat{\mathbf{x}}_0$ into the distribution $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$ and sample from it to get $\mathbf{x}_{t-1}$

Generative Modeling by Estimating Gradients of the Data Distribution

Yang Song, Stefano Ermon

July 12th, 2019

What is score matching?

If the data distribution is $p(\mathbf{x})$, then the score function is defined as

$$\nabla_{\mathbf{x}} \log p(\mathbf{x})$$

Note that if $p(\mathbf{x}) = \frac{e^{-f(\mathbf{x})}}{Z}$ (Z is our normalizing constant that makes density estimation intractable), then:

$$\nabla_{\mathbf{x}} \log p\,(\mathbf{x}) = -\nabla_{\mathbf{x}} f\,(\mathbf{x}) - \underbrace{\nabla_{\mathbf{x}} \log Z}_{=0} = -\nabla_{\mathbf{x}} f\,(\mathbf{x})$$

Don't need Z!

Modeling the score function → score-based model

$$\mathbf{s}_\theta(\mathbf{x}) \approx \log p(\mathbf{x})$$

Trained with the following objective:

$$\mathbb{E}_{p(\mathbf{x})}[\|\nabla_{\mathbf{x}} \log p(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x})\|_2^2]$$

Used for training energy-based models

Denoising score matching

Score matching of the perturbed distribution:

$$\mathcal{L}_{DSM} = \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}})}[\|\nabla_{\mathbf{x}} \log q_\sigma(\tilde{\mathbf{x}}) - \mathbf{s}_\theta(\tilde{\mathbf{x}})\|_2^2]$$

The following objective is equivalent!

$$\mathcal{L}_{DSM} = \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}},\mathbf{x})}[\|\nabla_{\mathbf{x}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) - \mathbf{s}_\theta(\tilde{\mathbf{x}})\|_2^2]$$

Since $\log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = -\frac{1}{2\sigma^2}(\tilde{\mathbf{x}} - \mathbf{x})^2$, then $\nabla_{\mathbf{x}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = -\frac{1}{\sigma^2}(\tilde{\mathbf{x}} - \mathbf{x})$

Final objective is:

$$\mathcal{L}_{DSM} = \mathbb{E}_{q_\sigma(\tilde{\mathbf{x}},\mathbf{x})}\left[\left\|\frac{1}{\sigma^2}(\tilde{\mathbf{x}} - \mathbf{x}) + \mathbf{s}_\theta(\tilde{\mathbf{x}})\right\|_2^2\right]$$

Tweedie's formula - optimal denoising function $f^*(\tilde{\mathbf{x}}) = \mathbf{x} \approx \tilde{\mathbf{x}} + \sigma^2 \nabla_{\tilde{\mathbf{x}}} \log p(\tilde{\mathbf{x}})$

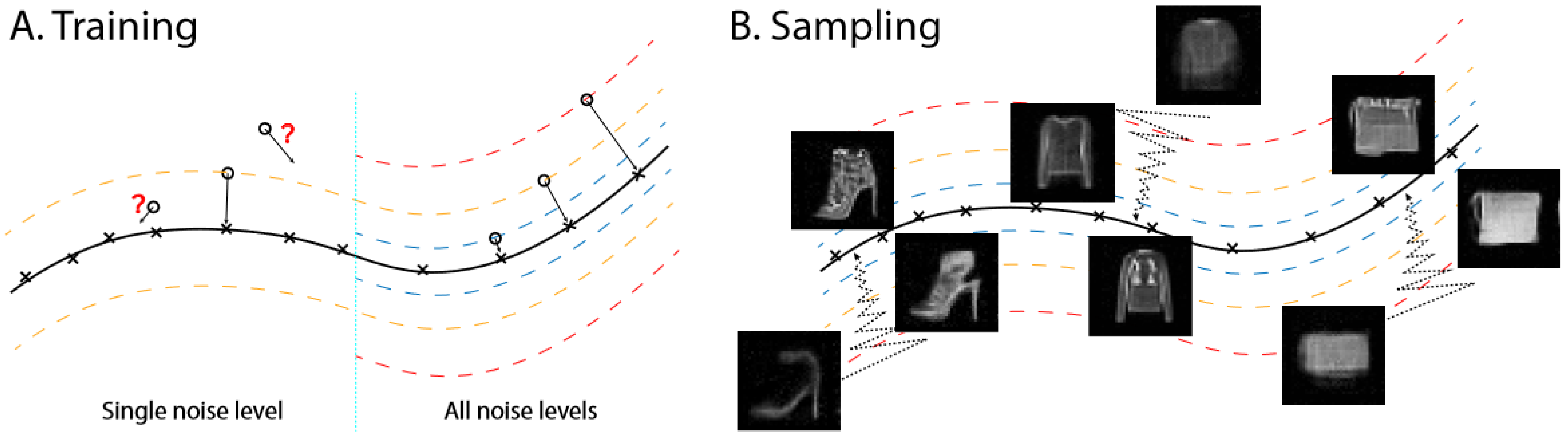
Denoising score matching written in DDPM notation

Denoising is equivalent to score matching:

$$\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t|\mathbf{x}_0) = -\frac{1}{(1-\bar{\alpha}_t)}(\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0) = -\frac{1}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}$$

$$\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) = -\sqrt{1-\bar{\alpha}_t}\,\mathbf{s}_\theta(\mathbf{x}_t, t)$$

Score matching perspective - Learning the data manifold!



Comparing Noise-conditional Score Networks (NCSN) and DDPM

DDPM noisy distribution:

$$q(\mathbf{x}_t|\mathbf{x}_0) \coloneqq \mathcal{N}\left(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}\right)$$

NCSN noisy distribution:

$$q(\mathbf{x}_t|\mathbf{x}_0) \coloneqq \mathcal{N}(\mathbf{x}_t;\, \mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

Sampling differences:

- DDPM decreases the noise scale at each Langevin step (ancestral sampling)
- NCSN runs multiple Langevin steps at each noise scale (annealed Langevin sampling)

Minor architectural differences

What’s next?

Time for a unified formulation!

We have discrete timesteps/noisescales, let’s try one of the oldest tricks in the (mathematician’s) book…

…take the limit and and make continuous!

Score-Based Generative Modeling through Stochastic Differential Equations

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, Ben Poole

Nov 26th, 2020

Ordinary Differential Equations (ODEs)

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}, t) \ \rightarrow \mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t$$

Solution:

$$\mathbf{x}(t) = \mathbf{x}(0) + \int_0^t \mathbf{f}(\mathbf{x}, \tau)\mathrm{d}\tau$$

Numerical solution:

$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t), t)\Delta t$$

Stochastic Differential Equations (SDEs)

Like ODEs… *but with noise!*

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \underbrace{\mathbf{f}(\mathbf{x}, t)}_{\text{drift coefficient}} + \underbrace{g(t)}_{\text{diffusion coefficient}} \frac{\mathrm{d}\mathbf{w}}{\mathrm{d}t} \rightarrow \mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}$$

Wiener process
(Brownian motion)

Numerical solution:

$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t), t)\Delta t + g(t)\mathbf{z}_t$$

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

SDE formulation of score-based generative modeling

In the limit of $\Delta t \to 0$, our diffusion process can now be described by $\mathbf{x}(t)$, indexed by a continuous time variable $t \in [0, T]$.

$\mathbf{x}(0) \sim p_0(x)$ which is the data distribution

$\mathbf{x}(T) \sim p_T(x)$ which is the prior distribution

Let's denote the probability density of $\mathbf{x}(t)$ as $p_t(\mathbf{x})$ and the transition kernel as $p_{st}(\mathbf{x}(t)|\mathbf{x}(s))$

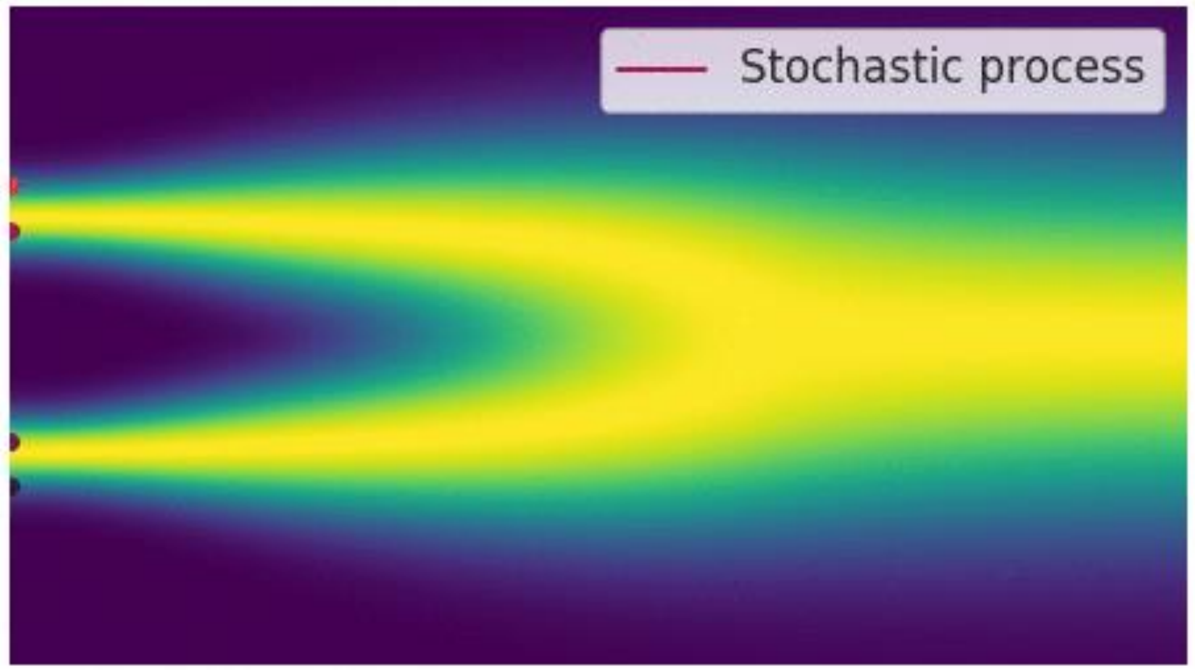
An important property

The reverse of a diffusion process is also a diffusion process, described by what is known as the Anderson reverse-time SDE:

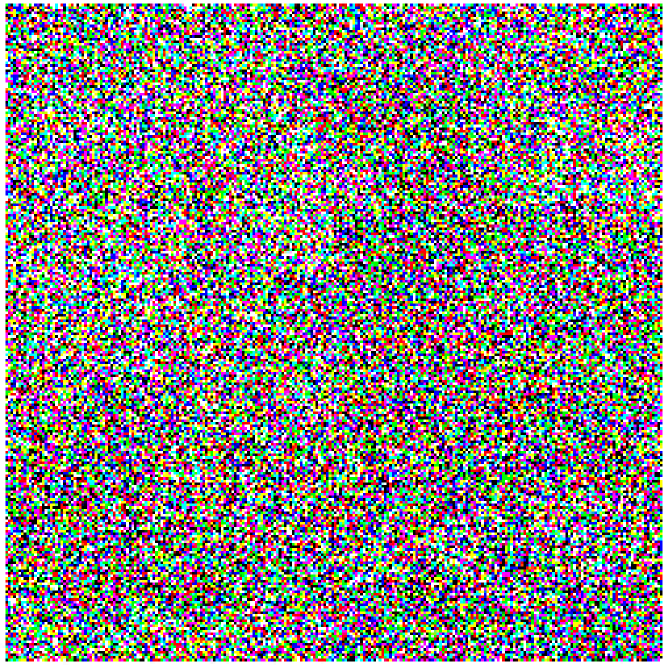
$$\mathrm{d}\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - g^2(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x})]\mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}}$$

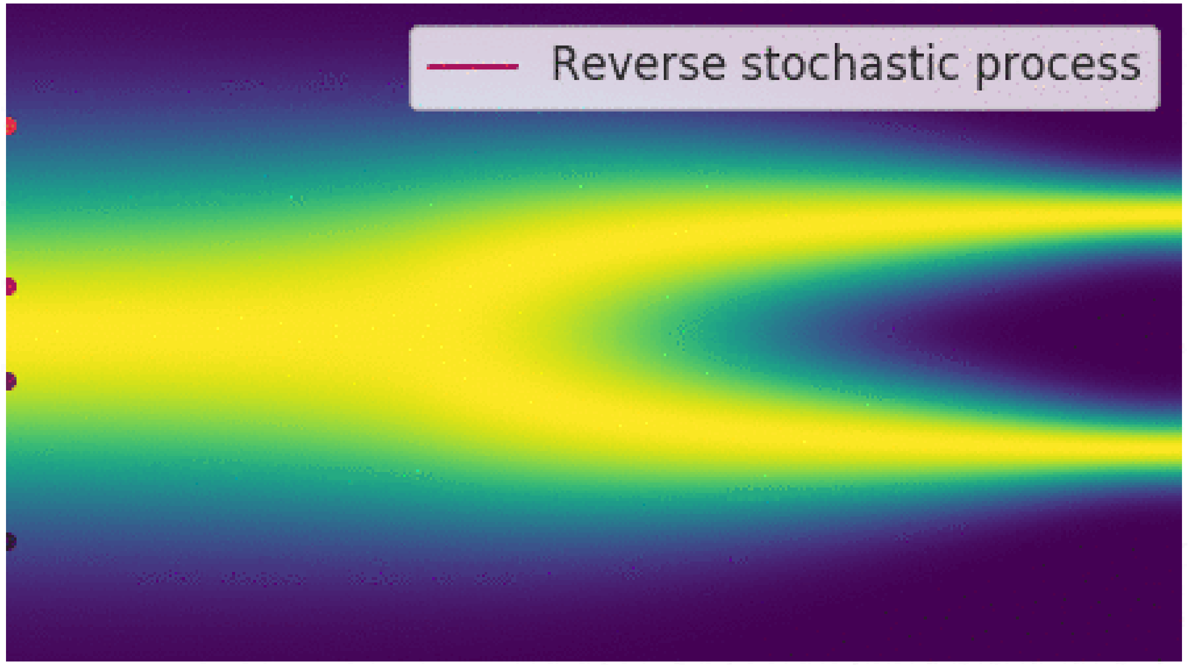
where $\bar{\mathbf{w}}$ is the Wiener process for time flowing backward from T to 0

Forward process as modeled by an SDE



Reverse process as modeled by a reverse-time SDE





Different score SDEs

NSCN (VE SDE):

$$\mathrm{d}\mathbf{x} = \sqrt{\frac{\mathrm{d}[\sigma^2(t)]}{\mathrm{d}t}}\,\mathrm{d}\mathbf{w}$$

DDPM (VP SDE):

$$\mathrm{d}\mathbf{x} = -\frac{1}{2}\beta(t)\mathbf{x}\,\mathrm{d}t + \sqrt{\beta(t)}\mathrm{d}\mathbf{w}$$

Sub-VP SDE:

$$\mathrm{d}\mathbf{x} = -\frac{1}{2}\beta(t)\mathbf{x}\,\mathrm{d}t + \sqrt{\beta(t)\left(1 - e^{-2\int_0^t \beta(s)\mathrm{d}s}\right)}\mathrm{d}\mathbf{w}$$

Training the score function for the reverse SDE

The training objective with denoising score matching is as follows:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\arg\min}\, \mathbb{E}_t \Big\{ \lambda(t) \mathbb{E}_{\mathbf{x}(0)} \mathbb{E}_{\mathbf{x}(t) \mid \mathbf{x}(0)} \Big[\big\| \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}(t), t) - \nabla_{\mathbf{x}(t)} \log p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0)) \big\|_2^2 \Big] \Big\}.$$

$$p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0)) = \begin{cases} \mathcal{N}\big(\mathbf{x}(t); \mathbf{x}(0), [\sigma^2(t) - \sigma^2(0)]\mathbf{I}\big), & \text{(VE SDE)} \\ \mathcal{N}\big(\mathbf{x}(t); \mathbf{x}(0) e^{-\frac{1}{2}\int_0^t \beta(s) \mathrm{d}s}, \mathbf{I} - \mathbf{I} e^{-\int_0^t \beta(s) \mathrm{d}s}\big) & \text{(VP SDE)} \\ \mathcal{N}\big(\mathbf{x}(t); \mathbf{x}(0) e^{-\frac{1}{2}\int_0^t \beta(s) \mathrm{d}s}, [1 - e^{-\int_0^t \beta(s) \mathrm{d}s}]^2 \mathbf{I}\big) & \text{(sub-VP SDE)} \end{cases}.$$

Solving the Reverse SDE

Given the SDE:

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

We have the reverse-time SDE:

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

Which we discretize as such:

$$\mathbf{x}_i = \mathbf{x}_{i+1} - f_{i+1}(\mathbf{x}_{i+1}) + (g_{i+1})^2 \boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x}_{i+1}, i+1) + g_{i+1}\mathbf{z}_{i+1}$$

DDPM ancestral sampling works out to be quite similar to this discretization for VP SDE.

Probability Flow ODEs

Given the SDE:

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

A corresponding "probability flow ODE" can be found:

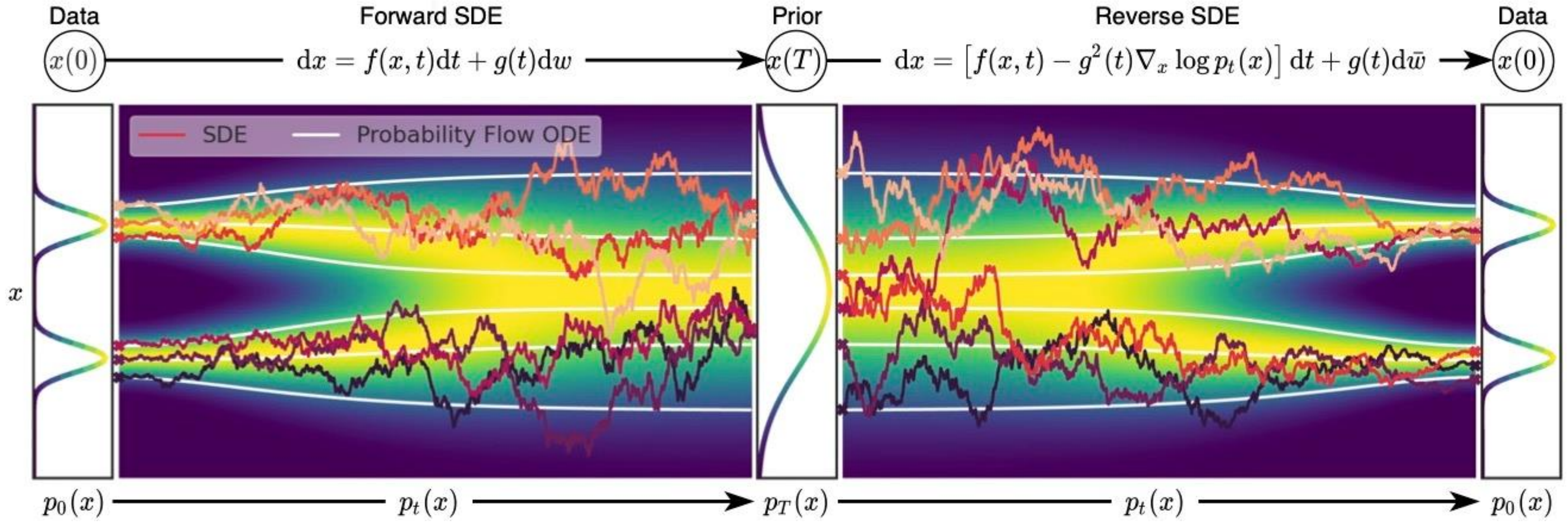
$$\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - \frac{1}{2} g^2(t) \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right] \mathrm{d}t$$

Describes the evolution of the probability distribution over time

We can numerically solve the ODE in reverse to generate samples!

Connection to neural ODEs/continuous normalizing flows enable exact log-likelihood calculation

Probability flow ODEs


Data
Forward SDE
Prior
Reverse SDE
Data
$x(0)$
$\mathrm{d}x = f(x,t)\mathrm{d}t + g(t)\mathrm{d}w$
$x(T)$
$\mathrm{d}x = \left[f(x,t) - g^2(t)\nabla_x \log p_t(x)\right]\mathrm{d}t + g(t)\mathrm{d}\bar{w}$
$x(0)$
SDE
Probability Flow ODE
x
$p_0(x)$
$p_t(x)$
$p_T(x)$
$p_t(x)$
$p_0(x)$

Uniquely identifiable encodings

By integrating over the ODE, we can map an input image $\mathbf{x}(0)$ to a uniquely identifiable encoding $\mathbf{x}(T)$

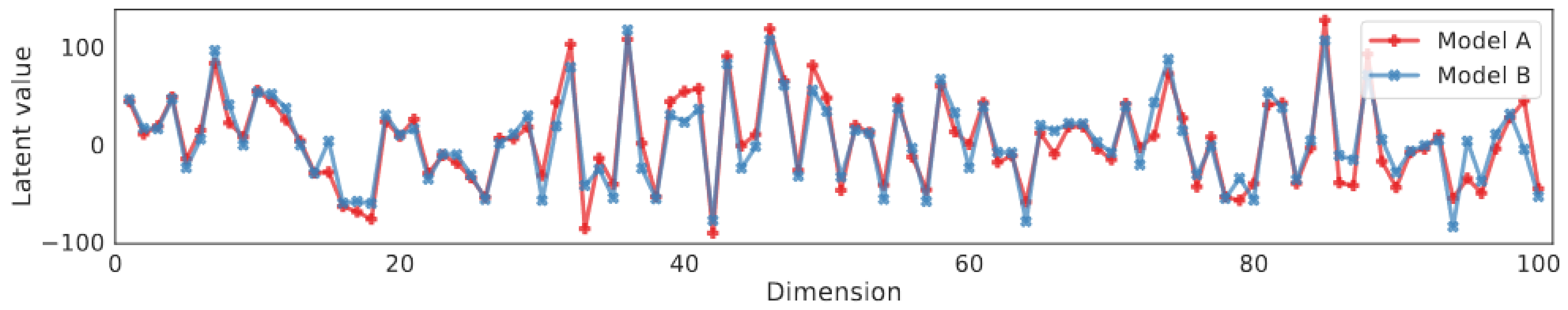
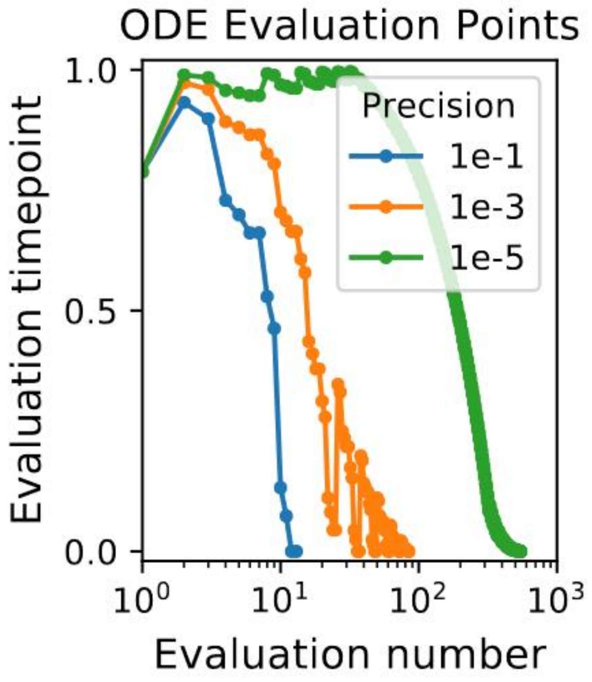
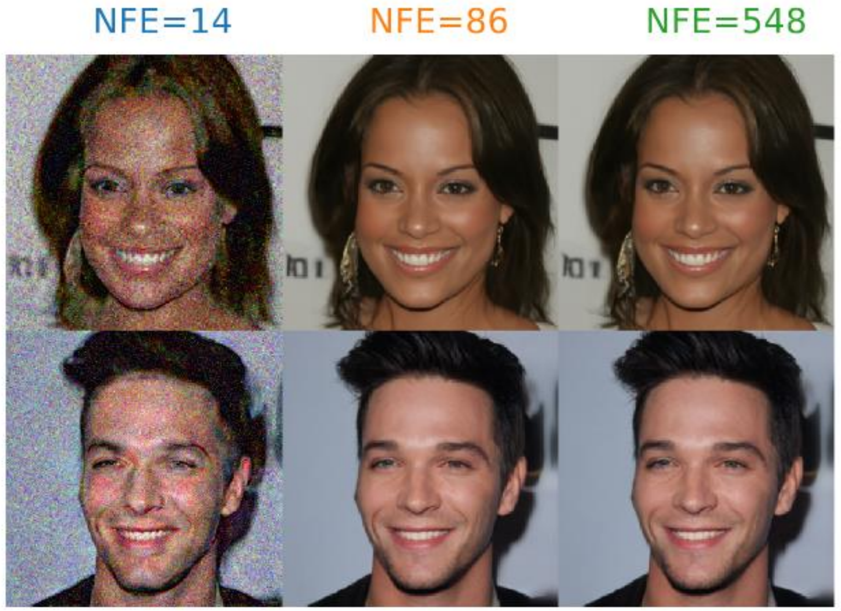


Figure 7: Comparing the first 100 dimensions of the latent code obtained for a random CIFAR-10 image. "Model A" and "Model B" are separately trained with different architectures.

ODE solvers enable faster sampling and controllable error tolerance





Denoising Diffusion Implicit Models

Jiaming Song, Chenlin Meng, Stefano Ermon

Oct 6th, 2020

Reviewing DDPM yet again!

Notation difference

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}\right) \rightarrow$$
$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{\alpha_t/\alpha_{t-1}}\mathbf{x}_{t-1}, (1-\alpha_t/\alpha_{t-1})\mathbf{I}\right)$$

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{\overline{\alpha}_t}\mathbf{x}_0, (1-\overline{\alpha}_t)\mathbf{I}\right) \rightarrow \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_0, (1-\alpha_t)\mathbf{I})$$

$$\mathbf{x}_t(\mathbf{x}_0, \epsilon) = \sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\epsilon, \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

Let us analyze the loss function

Our loss function:

$$L_\gamma(\epsilon_\theta) := \sum_{t=1}^{T} \gamma_t \mathbb{E}_{\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0), \epsilon_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})} \left[\left\| \epsilon_\theta^{(t)}(\sqrt{\alpha_t}\boldsymbol{x}_0 + \sqrt{1-\alpha_t}\epsilon_t) - \epsilon_t \right\|_2^2 \right]$$

We can see it only depends on $q(\mathbf{x}_t|\mathbf{x}_0)$ and not directly on $q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)$

So we can try to find inference processes (non-Markovian) that use the same marginal distribution!

Non-Markovian forward processes

A family of joint distributions indexed by a real vector σ:

$$q_\sigma(\mathbf{x}_{1:T} \mid \mathbf{x}_0) := q_\sigma(\mathbf{x}_T \mid \mathbf{x}_0) \prod_{t=2}^{T} q_\sigma(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$$

$$q_\sigma(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$$
$$= \mathcal{N}\left(\sqrt{\alpha_{t-1}}\mathbf{x}_0 + \sqrt{1-\alpha_{t-1}-\sigma_t^2}\,\frac{\mathbf{x}_t - \sqrt{\alpha_t}\mathbf{x}_0}{\sqrt{1-\alpha_t}}, \sigma^2\mathbf{I}\right)$$

This form is chosen since it ensures $q_\sigma(\mathbf{x}_T \mid \mathbf{x}_0) = \mathcal{N}\left(\sqrt{\alpha_T}\mathbf{x}_0, (1 - \alpha_T)\mathbf{I}\right)$

Non-Markovian forward processes

The forward process is therefore given by Bayes' rule:

$$q_\sigma(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{x}_0) = \frac{q_\sigma(\mathbf{x}_{t-1} \mid \mathbf{x}_t,\mathbf{x}_0)q_\sigma(\mathbf{x}_t|\mathbf{x}_0)}{q_\sigma(\mathbf{x}_{t-1}|\mathbf{x}_0)}$$

Note that the forward process is no longer Markovian. Additionally, note that the magnitude of σ controls how stochastic the process is. It becomes completely deterministic at $\sigma = 0$.

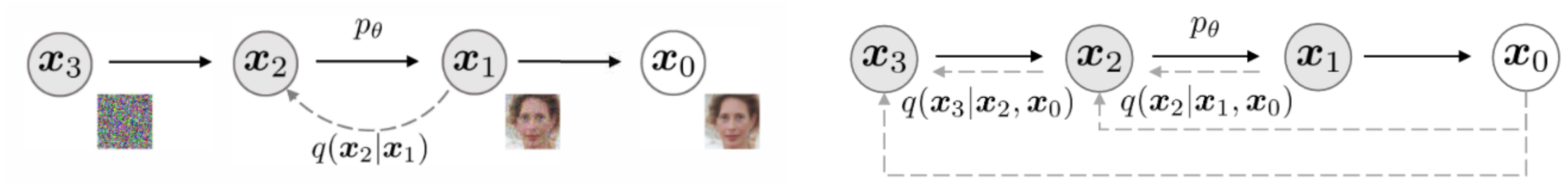


Figure 1: Graphical models for diffusion (left) and non-Markovian (right) inference models.

Key insight

If we want to train our reverse process model $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, we get the following objective:

$$J_\sigma(\epsilon_\theta) := \mathbb{E}_{\boldsymbol{x}_{0:T}\sim q_\sigma(\boldsymbol{x}_{0:T})}[\log q_\sigma(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) - \log p_\theta(\boldsymbol{x}_{0:T})] \qquad (11)$$

$$= \mathbb{E}_{\boldsymbol{x}_{0:T}\sim q_\sigma(\boldsymbol{x}_{0:T})}\left[\log q_\sigma(\boldsymbol{x}_T|\boldsymbol{x}_0) + \sum_{t=2}^{T}\log q_\sigma(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0) - \sum_{t=1}^{T}\log p_\theta^{(t)}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) - \log p_\theta(\boldsymbol{x}_T)\right]$$

However it can be shown that if the model is time-dependent, minimizing this objective is equivalent to minimizing the simplified DDPM objective!

DDIM is simply a novel sampling process for diffusion models!

DDIM sampling

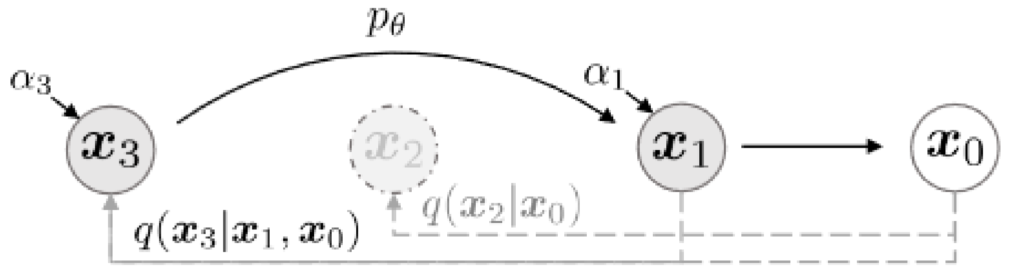
Sampling procedure:

$$\mathbf{x}_{t-1} = \sqrt{\alpha_{t-1}}\left(\frac{\mathbf{x}_t - \sqrt{1-\alpha_t}\epsilon_\theta(\mathbf{x}_t, t)}{\sqrt{\alpha_t}}\right) + \sqrt{1-\alpha_{t-1}-\sigma_t^2}\epsilon_\theta(\mathbf{x}_t, t) + \sigma_t\epsilon_t$$

where $\epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

Specific parameterization of σ_t gives DDPM and $\sigma_t = 0$ gives DDIM.

Accelerated generation:



Continuous version of DDIM

Rewrite the previous deterministic iterative procedure as such:

$$\frac{\boldsymbol{x}_{t-\Delta t}}{\sqrt{\alpha_{t-\Delta t}}} = \frac{\boldsymbol{x}_t}{\sqrt{\alpha_t}} + \left(\sqrt{\frac{1-\alpha_{t-\Delta t}}{\alpha_{t-\Delta t}}} - \sqrt{\frac{1-\alpha_t}{\alpha_t}}\right) \epsilon_\theta^{(t)}(\boldsymbol{x}_t)$$

Setting $\left(\sqrt{1-\alpha}/\alpha\right) = \sigma$ and $\boldsymbol{x}/\sqrt{\alpha} = \bar{\boldsymbol{x}}$, this is an Euler integration of the following ODE:

$$\mathrm{d}\bar{\boldsymbol{x}}(t) = \epsilon_\theta^{(t)}\left(\frac{\bar{\boldsymbol{x}}(t)}{\sqrt{\sigma^2+1}}\right)\mathrm{d}\sigma(t),$$

Continuous version of DDIM

The ODE is equivalent to Song et al.'s probability flow ODE for the VE SDE, but their discretization is not equivalent:

$$\frac{\boldsymbol{x}_{t-\Delta t}}{\sqrt{\alpha_{t-\Delta t}}} = \frac{\boldsymbol{x}_t}{\sqrt{\alpha_t}} + \frac{1}{2}\left(\frac{1-\alpha_{t-\Delta t}}{\alpha_{t-\Delta t}} - \frac{1-\alpha_t}{\alpha_t}\right) \cdot \sqrt{\frac{\alpha_t}{1-\alpha_t}} \cdot \epsilon_\theta^{(t)}(\boldsymbol{x}_t)$$

Just like Song et al.'s ODEs, DDIM ODE provides unique identifiable encodings

Improved Denoising Diffusion Probabilistic Models

Alex Nichol, Prafulla Dhariwal

Feb 18th, 2021

Importance of log-likelihood

Original DDPM was unable to achieve competitive log-likelihoods

This is an important metric that may indicate other issues like mode coverage/diversity

This paper systematically tries to improve log-likelihood

A simple change: $T = 1000 \to 4000$, changes log likelihood from 3.99 bits/dim to 3.77

Learned reverse process variance

The model outputs a vector v, which is used to give the following variance:

$$\Sigma_\theta(\mathbf{x}_t, t) = \exp\left(v \log \beta_t + (1 - v) \log \tilde{\beta}_t\right)$$

Since L_{simple} does not depend on the variance, a different reweighted objective is used:

$$L_{hybrid} = L_{simple} + \lambda L_{vlb}$$

Where $L_{vlb} = L_0 + L_1 + \cdots + L_{T-1} + L_T$ (KL terms that are tractable for Gaussians)

Other changes

A cosine noise schedule (for training)

$$\bar{\alpha}_t = \frac{f(t)}{f(0)}, \quad f(t) = \cos\left(\frac{t/T + s}{1 + s} \cdot \frac{\pi}{2}\right)^2$$

Importance sampling of L_{vlb} to improve training stability (gradient noise scale)

$$L_{\text{vlb}} = E_{t \sim p_t}\left[\frac{L_t}{p_t}\right], \text{ where } p_t \propto \sqrt{E[L_t^2]} \text{ and } \sum p_t = 1$$

Speeding up sampling

Given the original sequence of timesteps $(1, 2, \ldots, T)$, choose some arbitrary subsequence S. Given the original noise schedule $\bar{\alpha}_t$, we have $\bar{\alpha}_{S_t}$ for the subsequence, and also the following sampling variances:

$$\beta_{S_t} = 1 - \frac{\bar{\alpha}_{S_t}}{\bar{\alpha}_{S_{t-1}}}, \tilde{\beta}_{S_t} = \frac{1 - \bar{\alpha}_{S_{t-1}}}{1 - \bar{\alpha}_{S_t}} \beta_{S_t}$$

which can then be plugged into $\Sigma_\theta(\mathbf{x}_{S_t}, S_t)$, allowing for sampling from $p_\theta(\mathbf{x}_{S_{t-1}} \mid \mathbf{x}_{S_t})$

Improved performance with learned variances over fixed variances.
Near-optimal FIDs with 100 sampling steps

DDIM performs better with <50 steps but worse with >50 steps

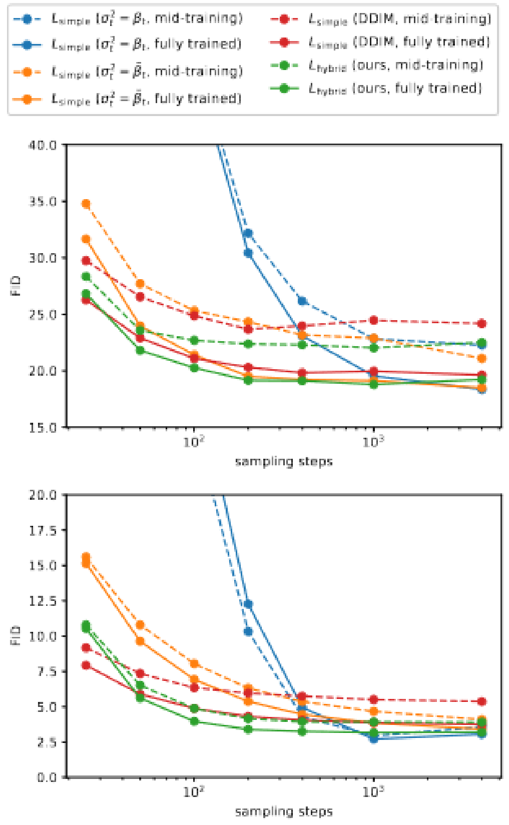


Figure 8. FID versus number of sampling steps, for models trained on ImageNet 64×64 (top) and CIFAR-10 (bottom). All models were trained with 4000 diffusion steps.

Class-conditional DDPM

Done to compare to class-conditional GANs

Class information is passed through the same pathway as the timesteps

A class embedding is added to the timestep embedding, which are passed to the residual blocks throughout the model.

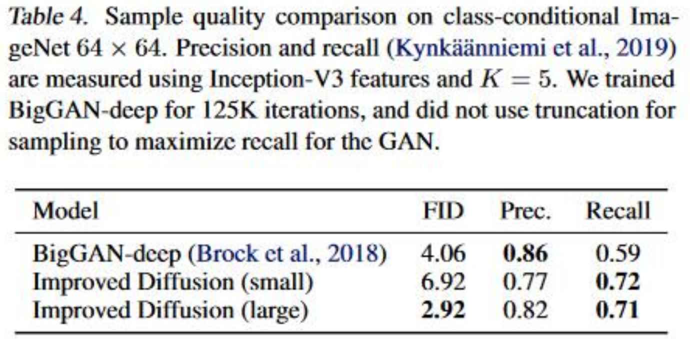


Table 4. Sample quality comparison on class-conditional ImageNet 64×64. Precision and recall (Kynkäänniemi et al., 2019) are measured using Inception-V3 features and $K = 5$. We trained BigGAN-deep for 125K iterations, and did not use truncation for sampling to maximize recall for the GAN.

Model	FID	Prec.	Recall
BigGAN-deep (Brock et al., 2018)	4.06	**0.86**	0.59
Improved Diffusion (small)	6.92	0.77	**0.72**
Improved Diffusion (large)	**2.92**	0.82	**0.71**

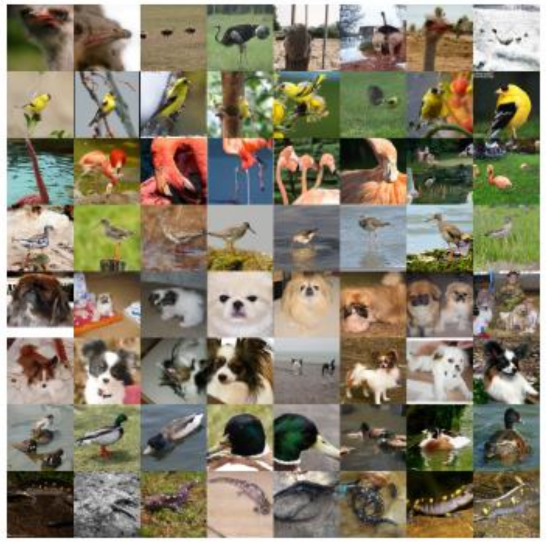


Figure 9. Class-conditional ImageNet 64×64 samples generated using 250 sampling steps from L_{hybrid} model (FID 2.92). The classes are 9: ostrich, 11: goldfinch, 130: flamingo, 141: redshank, 154: pekinese, 157: papillon, 97: drake and 28: spotted salamander. We see that there is a high diversity in each class, suggesting good coverage of the target distribution

Model Scaling

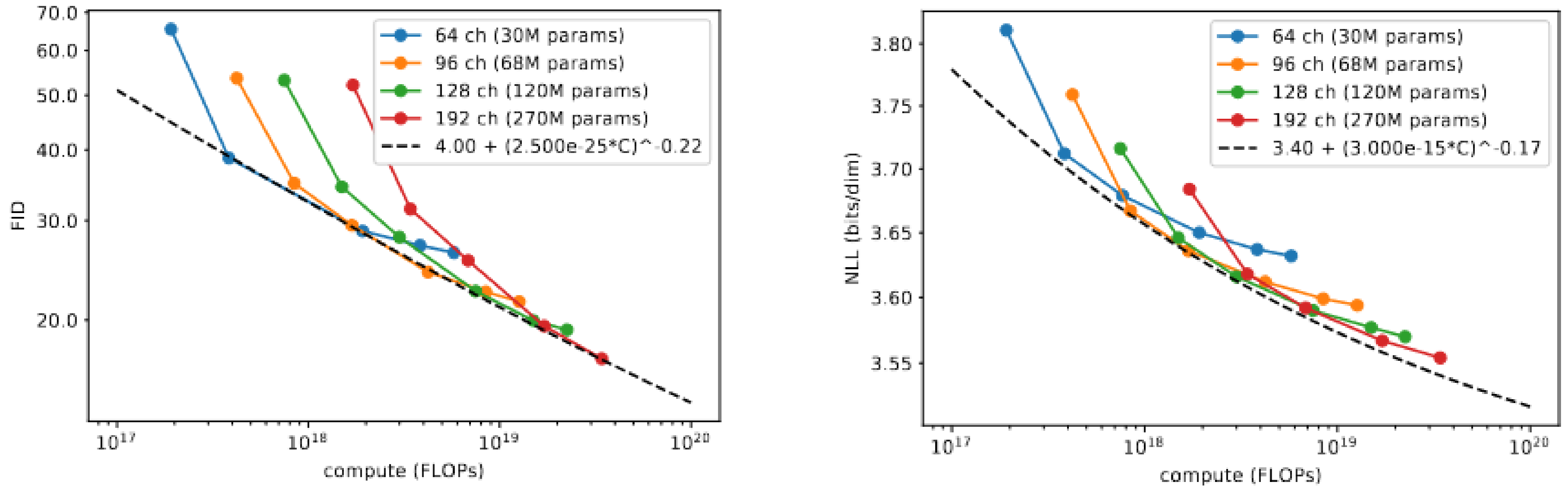


Figure 10. FID and validation NLL throughout training on ImageNet 64 × 64 for different model sizes. The constant for the FID trend line was approximated using the FID of in-distribution data. For the NLL trend line, the constant was approximated by rounding down the current state-of-the-art NLL (Roy et al., 2020) on this dataset.

Diffusion Models Beat GANs on Image Synthesis

Prafulla Dhariwal, Alex Nichol

May 11th, 2021

Architectural Improvements

Timestep+label embeddings are incorporated through shift and scaling of the group normalization

$$AdaGN(h, y) = y_s \text{GroupNorm}(h) + y_b$$

where h are the intermediate activations of a residual block, and $y = [y_s, y_b]$ are obtained from a linear projection of the embeddings

Ablated Diffusion Model (ADM):

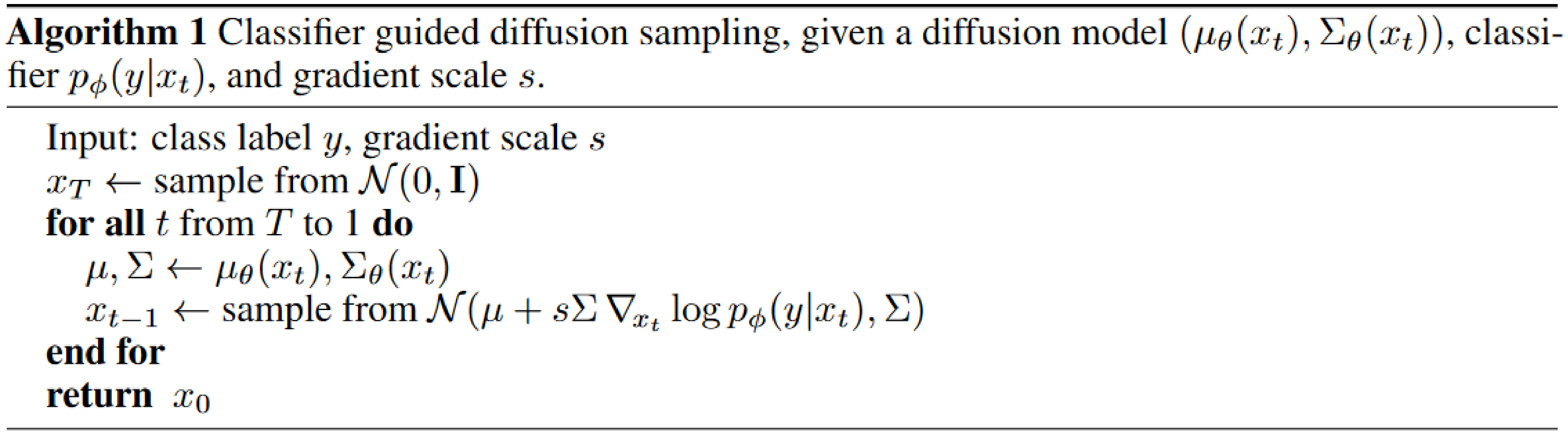
- Variable width with 2 residual blocks per resolution
- multiple heads with 64 channels per head
- attention at 32, 16 and 8 resolutions
- BigGAN residual blocks for up and downsampling
- AdaGN for injecting timestep+label embeddings into residual blocks.

Classifier Guidance

Mathematical derivation in paper demonstrates that the mean for the reverse process can be updated to be:

$$\mu_y = \mu + \Sigma g$$

where $g = \nabla_{x_t} \log p_\phi(y|x_t)$ (the gradient of the classifier output w.r.t. the input image x_t)



Algorithm 1 Classifier guided diffusion sampling, given a diffusion model $(\mu_\theta(x_t), \Sigma_\theta(x_t))$, classifier $p_\phi(y|x_t)$, and gradient scale s.

```
Input: class label $y$, gradient scale $s$
$x_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$
for all $t$ from $T$ to 1 do
    $\mu, \Sigma \leftarrow \mu_\theta(x_t), \Sigma_\theta(x_t)$
    $x_{t-1} \leftarrow$ sample from $\mathcal{N}(\mu + s\Sigma \nabla_{x_t} \log p_\phi(y|x_t), \Sigma)$
end for
return $x_0$
```

Classifier Guidance

The score-based formulation allows us to easily modify the DDIM sampling for classifier guidance. Specifically:

$$\begin{aligned}\nabla_{x_t} \log(p_\theta(x_t)p_\phi(y|x_t)) &= \nabla_{x_t} \log p_\theta(x_t) + \nabla_{x_t} \log p_\phi(y|x_t) \\ &= -\frac{1}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t) + \nabla_{x_t} \log p_\phi(y|x_t)\end{aligned}$$

So we can derive an updated noise predictor function to use for classifier guidance:

$$\hat{\epsilon}(x_t) := \epsilon_\theta(x_t) - \sqrt{1-\bar{\alpha}_t}\,\nabla_{x_t} \log p_\phi(y|x_t)$$

Classifier details

Classifier is simply the downsampling trunk of the U-net with an attention pooling at the 8x8 layer to produce the final output.

Classifier is trained on the noisy images, along with random crops to reduce overfitting

Scaling the classifier gradient to above 1 tends to result in better results

- Higher fidelity but lower diversity

Classifier Guidance Results

With a high enough scale, a guided unconditional model can get quite close in FID to an unguided conditional model (but still worse than guided conditional)

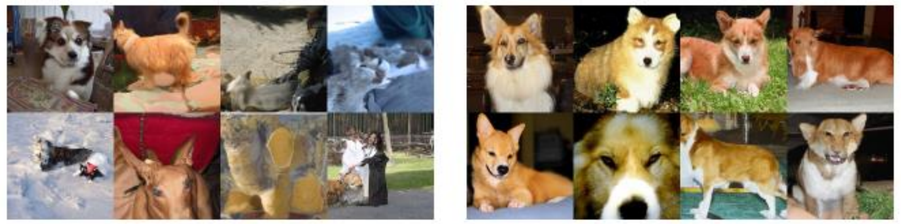


Figure 3: Samples from an unconditional diffusion model with classifier guidance to condition on the class "Pembroke Welsh corgi". Using classifier scale 1.0 (left; FID: 33.0) does not produce convincing samples in this class, whereas classifier scale 10.0 (right; FID: 12.0) produces much more class-consistent images.

Conditional	Guidance	Scale	FID	sFID	IS	Precision	Recall
✗	✗		26.21	**6.35**	39.70	0.61	0.63
✗	✓	1.0	33.03	6.99	32.92	0.56	**0.65**
✗	✓	10.0	**12.00**	10.40	**95.41**	**0.76**	0.44
✓	✗		10.94	6.02	100.98	0.69	**0.63**
✓	✓	1.0	**4.59**	**5.25**	186.70	0.82	0.52
✓	✓	10.0	9.11	10.93	**283.92**	**0.88**	0.32

Table 4: Effect of classifier guidance on sample quality. Both conditional and unconditional models were trained for 2M iterations on ImageNet 256×256 with batch size 256.

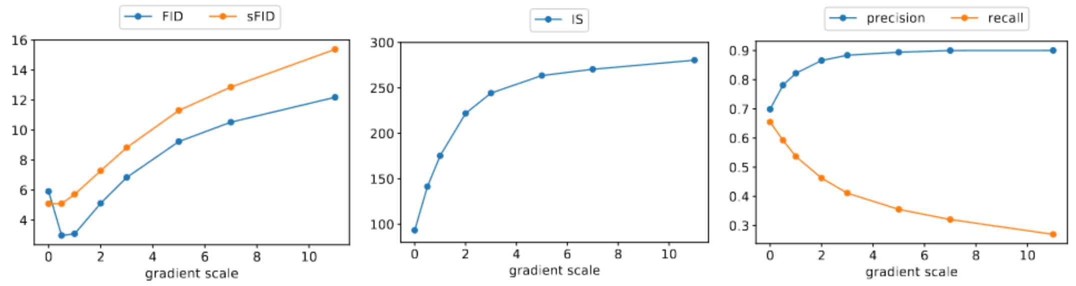
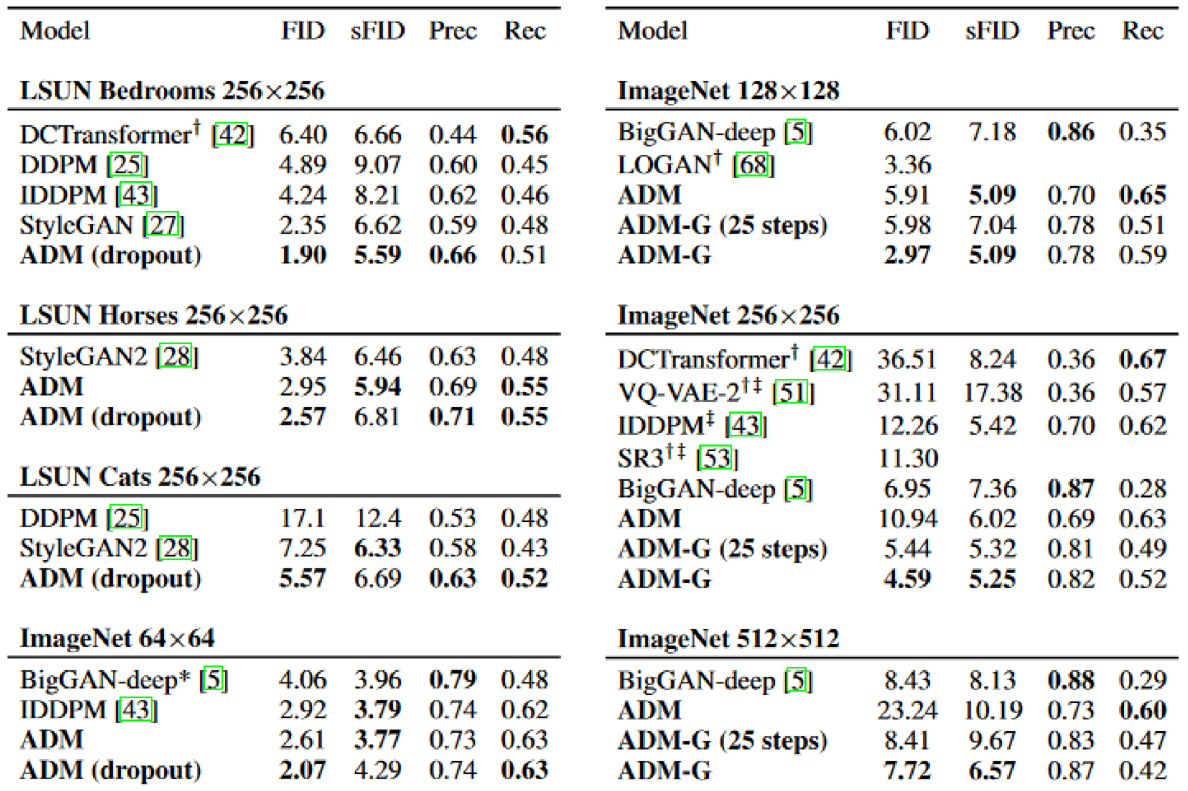


Figure 4: Change in sample quality as we vary scale of the classifier gradients for a class-conditional ImageNet 128×128 model.

Results



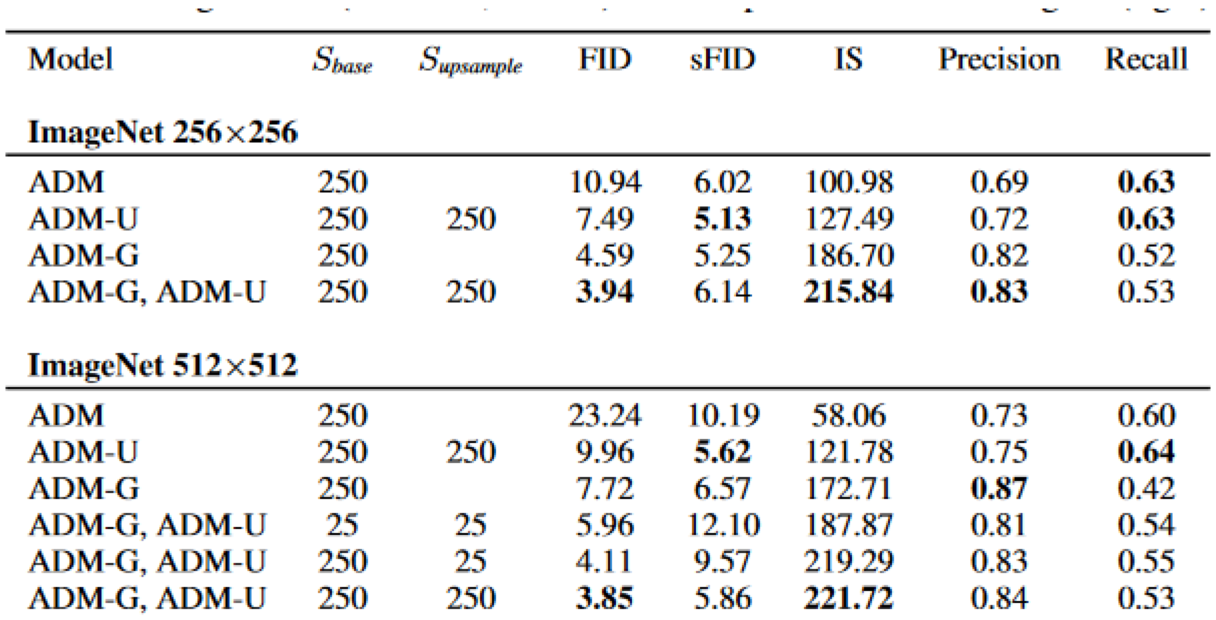
Model	FID	sFID	Prec	Rec
LSUN Bedrooms 256×256				
DCTransformer† [42]	6.40	6.66	0.44	**0.56**
DDPM [25]	4.89	9.07	0.60	0.45
IDDPM [43]	4.24	8.21	0.62	0.46
StyleGAN [27]	2.35	6.62	0.59	0.48
ADM (dropout)	**1.90**	**5.59**	**0.66**	0.51
LSUN Horses 256×256				
StyleGAN2 [28]	3.84	6.46	0.63	0.48
ADM	2.95	**5.94**	0.69	**0.55**
ADM (dropout)	**2.57**	6.81	**0.71**	**0.55**
LSUN Cats 256×256				
DDPM [25]	17.1	12.4	0.53	0.48
StyleGAN2 [28]	7.25	**6.33**	0.58	0.43
ADM (dropout)	**5.57**	6.69	**0.63**	**0.52**
ImageNet 64×64				
BigGAN-deep* [5]	4.06	3.96	**0.79**	0.48
IDDPM [43]	2.92	**3.79**	0.74	0.62
ADM	2.61	**3.77**	0.73	0.63
ADM (dropout)	**2.07**	4.29	0.74	**0.63**

Model	FID	sFID	Prec	Rec
ImageNet 128×128				
BigGAN-deep [5]	6.02	7.18	**0.86**	0.35
LOGAN† [68]	3.36			
ADM	5.91	**5.09**	0.70	**0.65**
ADM-G (25 steps)	5.98	7.04	0.78	0.51
ADM-G	**2.97**	**5.09**	0.78	0.59
ImageNet 256×256				
DCTransformer† [42]	36.51	8.24	0.36	**0.67**
VQ-VAE-2†‡ [51]	31.11	17.38	0.36	0.57
IDDPM‡ [43]	12.26	5.42	0.70	0.62
SR3†‡ [53]	11.30			
BigGAN-deep [5]	6.95	7.36	**0.87**	0.28
ADM	10.94	6.02	0.69	0.63
ADM-G (25 steps)	5.44	5.32	0.81	0.49
ADM-G	**4.59**	**5.25**	0.82	0.52
ImageNet 512×512				
BigGAN-deep [5]	8.43	8.13	**0.88**	0.29
ADM	23.24	10.19	0.73	**0.60**
ADM-G (25 steps)	8.41	9.67	0.83	0.47
ADM-G	**7.72**	**6.57**	0.87	0.42

Table 5: Sample quality comparison with state-of-the-art generative models for each task. ADM refers to our ablated diffusion model, and ADM-G additionally uses classifier guidance. LSUN diffusion models are sampled using 1000 steps (see Appendix J). ImageNet diffusion models are sampled using 250 steps, except when we use the DDIM sampler with 25 steps. *No BigGAN-deep model was available at this resolution, so we trained our own. †Values are taken from a previous paper, due to lack of public models or samples. ‡Results use two-resolution stacks.

Results

Two-stage pipeline (originally proposed in iDDPM, in appendix): guidance at lower resolution and then upsample gets best 512x512 results



Model	S_{base}	$S_{upsample}$	FID	sFID	IS	Precision	Recall
ImageNet 256×256							
ADM	250		10.94	6.02	100.98	0.69	**0.63**
ADM-U	250	250	7.49	**5.13**	127.49	0.72	**0.63**
ADM-G	250		4.59	5.25	186.70	0.82	0.52
ADM-G, ADM-U	250	250	**3.94**	6.14	**215.84**	**0.83**	0.53
ImageNet 512×512							
ADM	250		23.24	10.19	58.06	0.73	0.60
ADM-U	250	250	9.96	**5.62**	121.78	0.75	**0.64**
ADM-G	250		7.72	6.57	172.71	**0.87**	0.42
ADM-G, ADM-U	25	25	5.96	12.10	187.87	0.81	0.54
ADM-G, ADM-U	250	25	4.11	9.57	219.29	0.83	0.55
ADM-G, ADM-U	250	250	**3.85**	5.86	**221.72**	0.84	0.53

Results

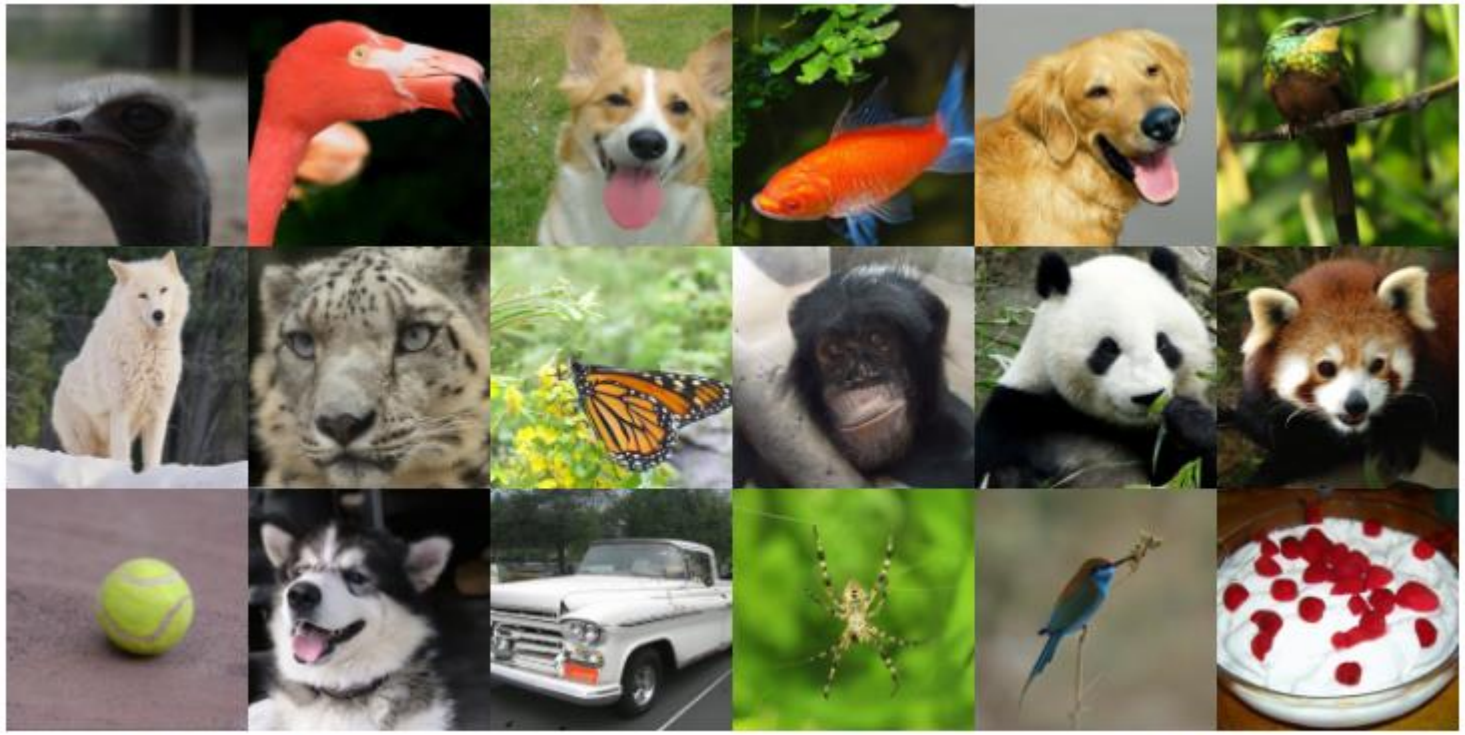


Figure 1: Selected samples from our best ImageNet 512×512 model (FID 3.85)

Summary and Big Picture

History of diffusion models

Diffusion models were originally invented in 2015 by Jascha Sohl-Dickstein et al. at Stanford.

DDPM by Jonathan Ho et al. at UC Berkeley greatly simplified diffusion models with high-quality results by connecting it to VAEs, transforming it into a simple denoising task and also pointed out connection to NCSNs.

NCSNs by Song and Ermon at Stanford explored how a denoising task at multiple noise levels can be used to efficiently and accurately learn gradients of a data distribution (score) and use that to sample from it. NCSN++ was an updated model with additional improvements.

History of diffusion models

Score SDEs and Probability Flow ODEs by Song et al. provided a continuous formulation that described NCSNs and DDPMs in a single framework. It also allowed the leveraging of SDE/ODE solvers for sampling.

DDIM by Song, Meng, and Ermon at Stanford provided a deterministic and accelerated sampling algorithm.

The OpenAI gang (Dhariwal and Nichol) explored improvements to DDPM with IDDPM (focused on log likelihood improvements) and ADM (a detailed ablation study of architectural changes)

How and why do diffusion models work?

In all cases we train some sort of model to undo corruptions to a data.
This can be thought of as bringing your datapoint back to the data manifold.

This becomes equivalent to following the gradient back to the manifold.

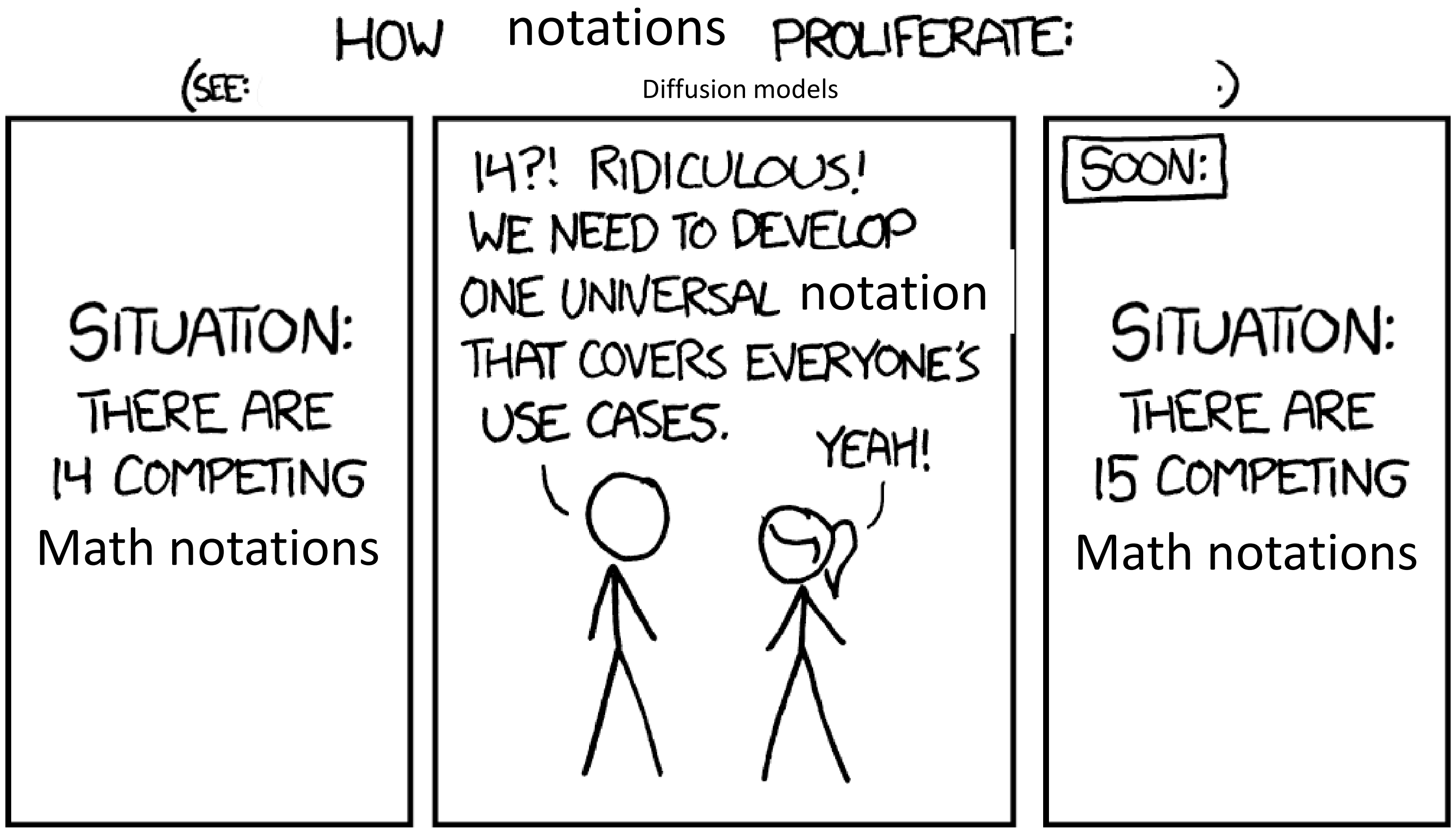
We train over a variety of noise levels to cover the entire data space and accurately estimate the gradients anywhere to bring us to the manifold.

Diffusion models are sampled in an iterative process with two steps per iteration:

1. First predict the denoised image
2. Add some noise to the predicted image to give us the next iteration, a slightly less noisy image

Classifier-Free Diffusion Guidance

Differences in notation!


HOW notations PROLIFERATE:
(SEE: Diffusion models.)
SITUATION: THERE ARE 14 COMPETING Math notations
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL notation THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING Math notations

Standard classifier guidance

Standard conditional diffusion model:

$$\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{c}) \approx -\sigma_t \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t | \mathbf{c})$$

Classifier guidance modifies the score function:

$$\begin{aligned}\tilde{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t, \mathbf{c}) &= \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{c}) - w\sigma_t \nabla_{\mathbf{x}_t} \log p(\mathbf{c} | \mathbf{x}_t) \\ &\approx -\sigma_t \nabla_{\mathbf{x}_t}[\log p(\mathbf{x}_t | \mathbf{c}) + w \log p(\mathbf{c} | \mathbf{x}_t)]\end{aligned}$$

where w is the guidance scale. This corresponds to a distribution of:

$$\tilde{p}_\theta(\mathbf{x}_t, \mathbf{c}) \propto p_\theta(\mathbf{x}_t|\mathbf{c}) p_\theta(\mathbf{c}|\mathbf{x}_t)^w$$

This boosts the probability of generating images that are correctly classified by classifier $p_\theta(\mathbf{c}|\mathbf{x}_t)$

Demonstration of guidance scale

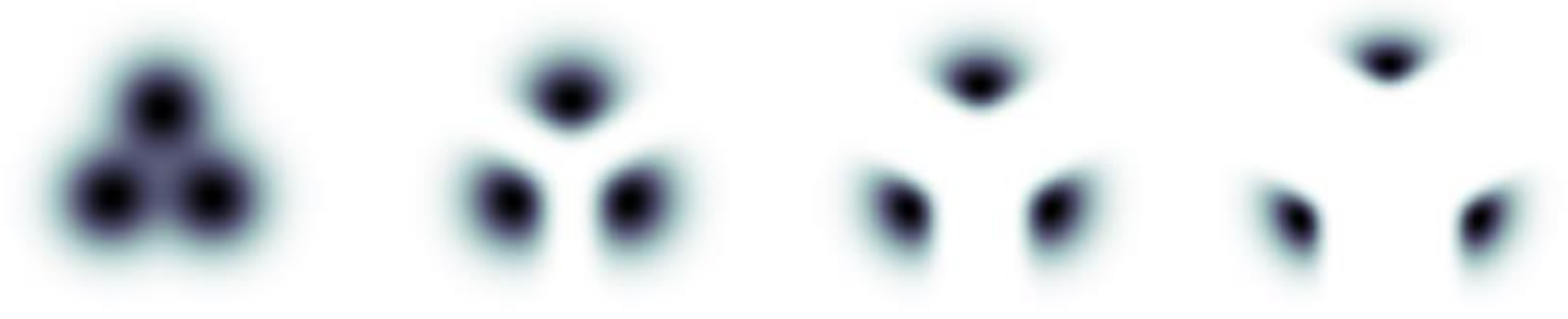


Figure 2: The effect of guidance on a mixture of three Gaussians, each mixture component representing data conditioned on a class. The leftmost plot is the non-guided marginal density. Left to right are densities of mixtures of normalized guided conditionals with increasing guidance strength.

Theoretical equivalence of unconditional and conditional models

Theoretically, applying classifier guidance with weight $w+1$ to an unconditional model should be equivalent to applying classifier guidance to a conditional model with weight w

$$\begin{aligned}&\boldsymbol{\epsilon}_\theta(\mathbf{x}_t) - (w+1)\sigma_t \nabla_{\mathbf{x}_t} \log p_\theta(\,\mathbf{c} \mid \mathbf{x}_t\,) \\ &\approx -\sigma_t \nabla_{\mathbf{x}_t}[\log p(\mathbf{x}_t) + (w+1)\log p_\theta(\,\mathbf{c} \mid \mathbf{x}_t\,)]\end{aligned}$$

Bayes' Rule tells us:

$$\log p\,(\,\mathbf{x}_t \mid \mathbf{c}\,) = \log p\,(\,\mathbf{c} \mid \mathbf{x}_t\,) + \log p\,(\mathbf{x}_t) - \log p\,(\mathbf{c})$$

Therefore:

$$\boldsymbol{\epsilon}_\theta(\mathbf{x}_t) - (w+1)\sigma_t \nabla_{\mathbf{x}_t} \log p_\theta(\,\mathbf{c} \mid \mathbf{x}_t\,) \approx -\sigma_t \nabla_{\mathbf{x}_t}[\log p(\mathbf{x}_t \mid \mathbf{c}) + w \log p(\mathbf{c} \mid \mathbf{x}_t)]$$

Classifier-Free Diffusion Guidance

A conditional model $p_\theta(\mathbf{x}_t|\mathbf{c})$ (parameterized by $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{c})$) and an unconditional model $p_\theta(\mathbf{x}_t)$(parameterized by $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t)$)

Can be represented by a *single* neural network $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{c})$ where $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t) = \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{c} = \emptyset)$

During training, randomly set $\mathbf{c}$ to $\emptyset$ with some probability p_{uncond} (a new hyperparam)

Classifier-Free Diffusion Guidance

Let's examine Bayes' Rule again:

$$\log p\,(\,\mathbf{x}_t \mid \mathbf{c}\,) = \log p\,(\,\mathbf{c} \mid \mathbf{x}_t\,) + \log p\,(\mathbf{x}_t) - \log p\,(\mathbf{c})$$
$$\log p\,(\,\mathbf{c} \mid \mathbf{x}_t\,) = \log p\,(\,\mathbf{x}_t \mid \mathbf{c}\,) + \log p\,(\mathbf{c}) - \log p\,(\mathbf{x}_t)$$
$$\nabla_{\mathbf{x}_t} \log p\,(\,\mathbf{c} \mid \mathbf{x}_t\,) = \nabla_{\mathbf{x}_t}[\log p\,(\,\mathbf{x}_t \mid \mathbf{c}\,) - \log p\,(\mathbf{x}_t)]$$

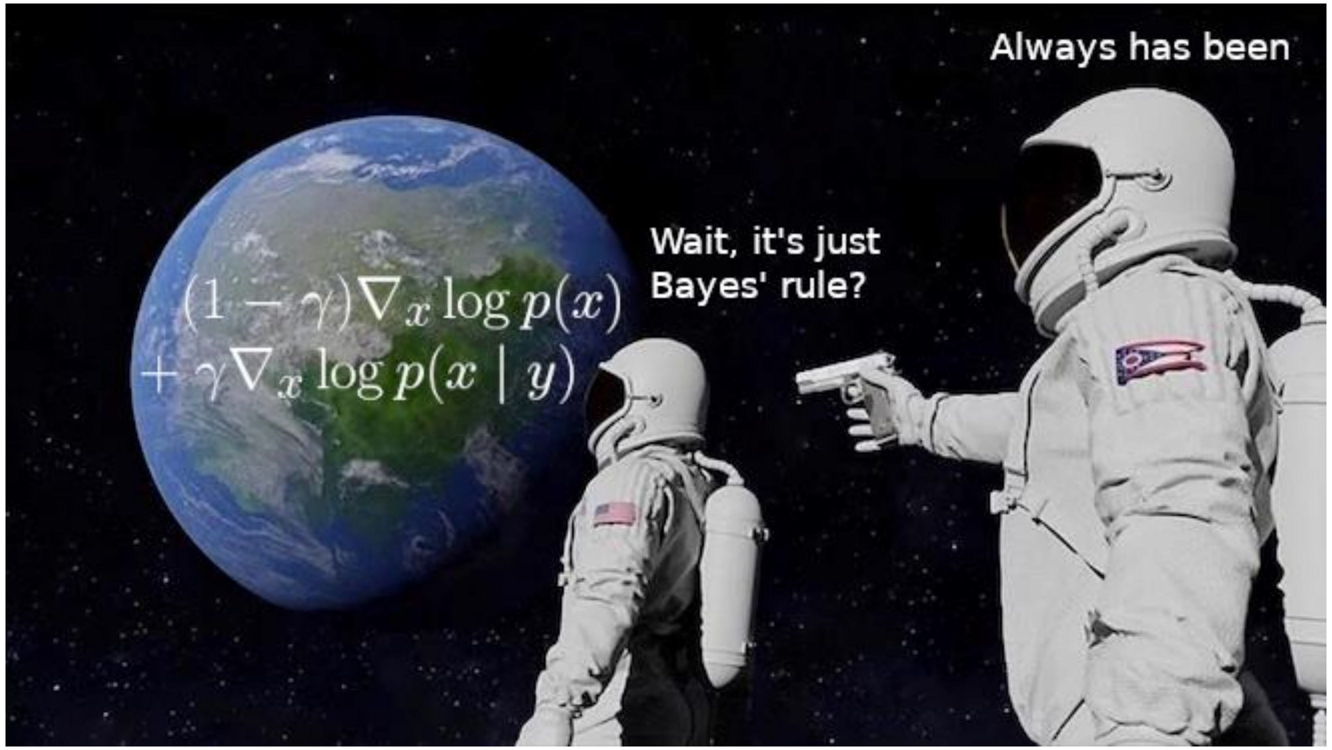
Use the gradient of this implicit classifier in classifier guidance!

$$\tilde{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t, \mathbf{c}) = \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{c}) - w\sigma_t \nabla_{\mathbf{x}_t} \log p(\mathbf{c} \mid \mathbf{x}_t)$$
$$= \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{c}) - w\sigma_t \nabla_{\mathbf{x}_t}[\log p\,(\,\mathbf{x}_t \mid \mathbf{c}\,) - \log p\,(\mathbf{x}_t)] = \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{c}) + w\,[\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{c}) - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t)]$$

Sampling is done with the following modified score function:

$$\tilde{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t, \mathbf{c}) = (1 + w)\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{c}) - w\boldsymbol{\epsilon}_\theta(\mathbf{x}_t)$$

It’s just Bayes’ Rule!


$(1-\gamma)\nabla_x \log p(x)$
$+ \gamma\nabla_x \log p(x \mid y)$
Wait, it's just Bayes' rule?
Always has been

Classifier-Free Guidance (CFG)

We have *constructed an implicit classifier* from our conditional generative model using Bayes' Rule and used that to guide the model.

Discriminative models tend to be better than implicit classifiers derived from generative models however…

Empirically classifier-free guidance works well though!

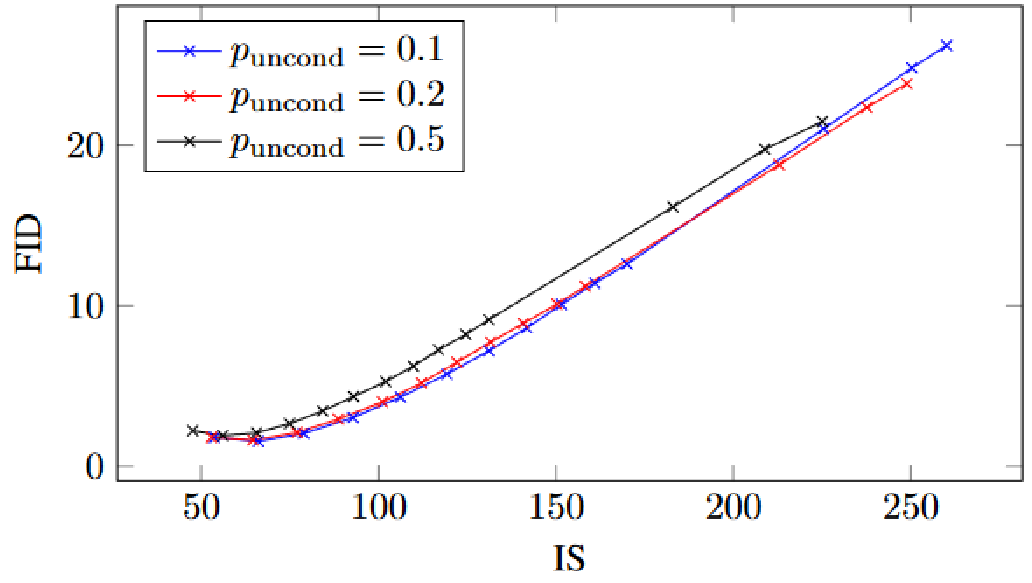
CFG decreases the unconditional likelihood of the sample while increasing the conditional likelihood.

Disadvantage: additional network evaluation needed for unconditional model for each sampling step

Results

Best FID results are obtained with small amount of guidance (w=0.1 or 0.3) and best Inception score is obtained with strong guidance (w >= 4)

Relatively small model capacity needed for unconditional generation



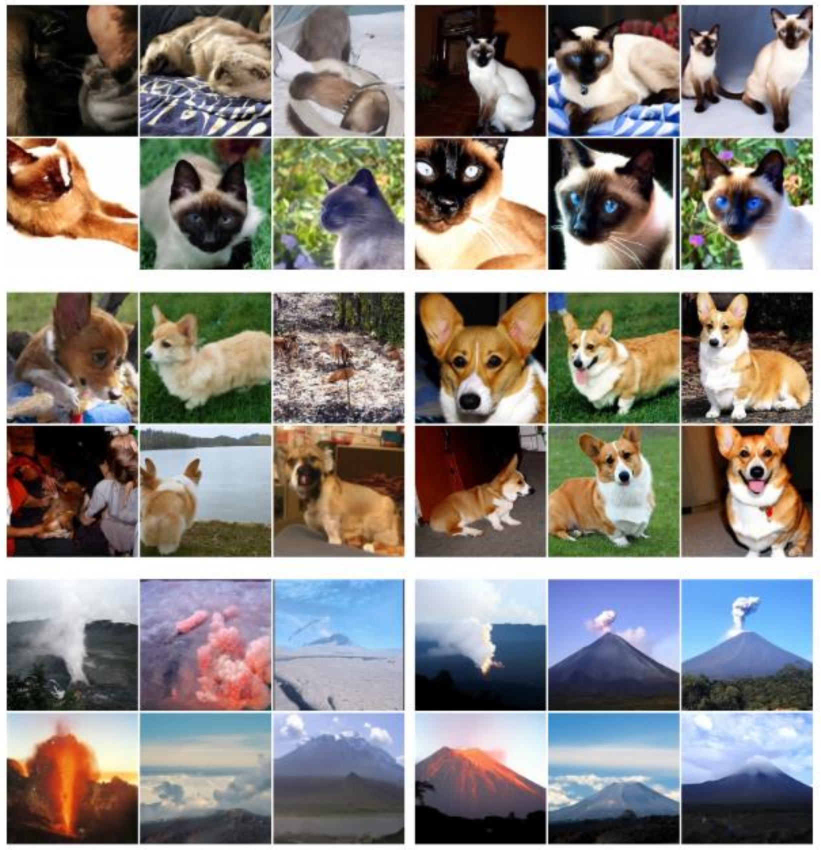
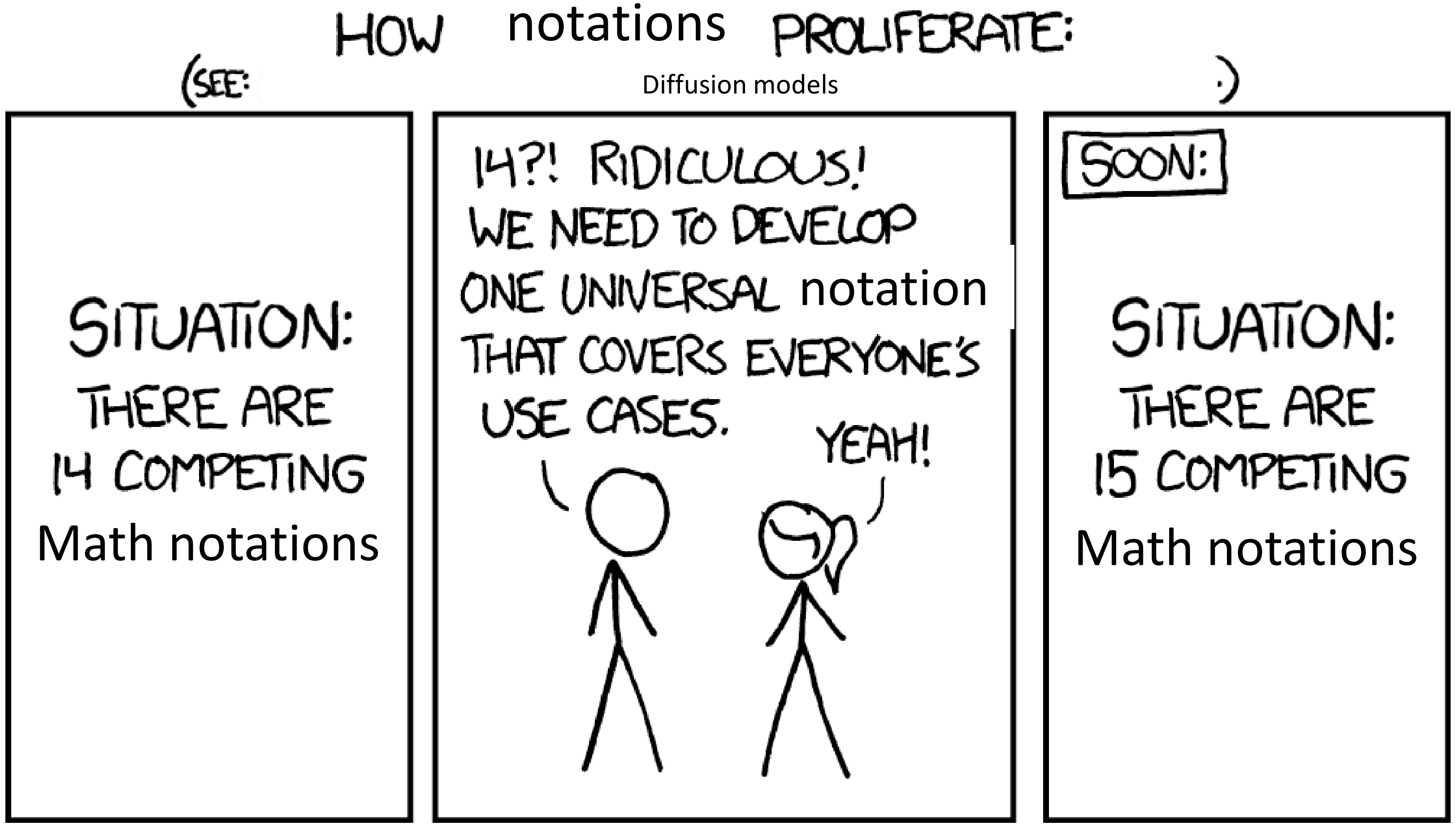


Figure 3: Classifier-free guidance on 128x128 ImageNet. Left: non-guided samples, right: classifier-free guided samples with $w = 3.0$. Interestingly, strongly guided samples such as these display saturated colors. See Fig. 8 for more.

Variational Diffusion Models

Differences in notation!


HOW notations PROLIFERATE:
(SEE: Diffusion models .)
SITUATION: THERE ARE 14 COMPETING Math notations
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL notation THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING Math notations

Forward process

Forward process runs between $t = 0$ and $t = 1$ and the latent variables of intermediate t is given by:

$$q(\mathbf{z}_t|\mathbf{x}) = \mathcal{N}(\alpha_t\mathbf{x}, \sigma_t^2\mathbf{I})$$

Signal-noise ratio:

$$\mathrm{SNR}(t) = \alpha_t^2/\sigma_t^2$$

where α_t and σ_t^2 are functions of t

Variance-preserving: $\alpha_t^2 = \sqrt{(1 - \sigma_t^2)}$

Variance-exploding: $\alpha_t^2 = 1$

Forward process

For $s < t$:

$$q(\mathbf{z}_t|\mathbf{z}_s) = \mathcal{N}\left(\alpha_{t|s}\mathbf{x}, \sigma_{t|s}^2\mathbf{I}\right)$$

where

$$\alpha_{t|s} = \alpha_t/\alpha_s,$$
$$\sigma_{t|s}^2 = \sigma_t^2 - \alpha_{t|s}^2\sigma_s^2$$

$$q(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}_Q(\mathbf{z}_t, \mathbf{x}; s, t), \sigma_Q^2(s,t)\mathbf{I})$$

where $\sigma_Q^{-2}(s,t) = \sigma_s^{-2} + \alpha_{t|s}^2\sigma_{t|s}^{-2} = \sigma_{t|s}^2\sigma_s^2/\sigma_t^2$

and $\boldsymbol{\mu}_Q(\mathbf{z}_t, \mathbf{x}; s, t) = \dfrac{\alpha_{t|s}\sigma_s^2}{\sigma_t^2}\mathbf{z}_t + \dfrac{\alpha_s\sigma_{t|s}^2}{\sigma_t^2}\mathbf{x}.$

Noise schedule

Noise schedule is learnt (parameterized by an MLP)!

$$\sigma_t^2 = \mathrm{sigmoid}(\gamma_\eta(t))$$

where $\gamma_\eta(t)$ is the neural network with params η

For variance-preserving diffusion processes:

$$\alpha_t^2 = \mathrm{sigmoid}(-\gamma_\eta(t))$$
$$\mathrm{SNR}(t) = \exp\left(-\gamma_\eta(t)\right)$$

Reverse-time generative model

Discretize time $t = 0 \rightarrow 1$ into T timesteps. $s(i) = (i-1)/T$ and $t(i) = i/T$. Our hierarchical generative model is:

$$p(\mathbf{x}) = \int_{\mathbf{z}} p(\mathbf{z}_1) p(\mathbf{x}|\mathbf{z}_0) \prod_{i=1}^{T} p(\mathbf{z}_{s(i)}|\mathbf{z}_{t(i)}).$$

The marginal distribution is a spherical Gaussian:

$$p(\mathbf{z}_1) = \mathcal{N}(\mathbf{z}_1; 0, \mathbf{I}).$$

The distribution $p(\mathbf{x}|\mathbf{z}_0)$ is given as:

$$p(\mathbf{x}|\mathbf{z}_0) = \prod_i p(x_i|z_{0,i}),$$

Reverse-time generative model

Conditional distribution:

$$p(\mathbf{z}_s|\mathbf{z}_t) = q(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x} = \hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t)),$$

where $\hat{\mathbf{x}}_\theta(\mathbf{z}_t; t)$ is the denoising model. It is parameterized in terms of a noise prediction model:

$$\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t) = (\mathbf{z}_t - \sigma_t \hat{\epsilon}_{\boldsymbol{\theta}}(\mathbf{z}_t; t))/\alpha_t.$$

where $\hat{\boldsymbol{\epsilon}}_\theta(\mathbf{z}_t; t)$ is a neural network

Fourier features

Append channels $\sin(2^n \pi \mathbf{z})$ and $\cos(2^n \pi \mathbf{z})$ where $n \in \{n_{min}, \dots, n_{max}\}$

These features amplify small changes to the input data, allowing the model to capture fine scale details of the data

Leads to large improvements in likelihood. Best results obtained with $n_{min} = 7, n_{max} = 8$

Variational lower bound

Treating it as a hierarchical generative model, like an HAVE, the model is trained with the standard VLB:

$$-\log p(\mathbf{x}) \leq -\text{VLB}(\mathbf{x}) = \underbrace{D_{KL}(q(\mathbf{z}_1|\mathbf{x})||p(\mathbf{z}_1))}_{\text{Prior loss}} + \underbrace{\mathbb{E}_{q(\mathbf{z}_0|\mathbf{x})}\left[-\log p(\mathbf{x}|\mathbf{z}_0)\right]}_{\text{Reconstruction loss}} + \underbrace{\mathcal{L}_T(\mathbf{x})}_{\text{Diffusion loss}}.$$

We now will investigate the diffusion loss further

Discrete-time model

In the case of finite timesteps, we get the following loss function:

$$\mathcal{L}_T(\mathbf{x}) = \sum_{i=1}^{T} \mathbb{E}_{q(\mathbf{z}_{t(i)}|\mathbf{x})} D_{KL}[q(\mathbf{z}_{s(i)}|\mathbf{z}_{t(i)}, \mathbf{x})||p(\mathbf{z}_{s(i)}|\mathbf{z}_{t(i)})].$$

This simplifies significantly to:

$$\mathcal{L}_T(\mathbf{x}) = \frac{T}{2} \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I}), i\sim U\{1,T\}} \left[(\text{SNR}(s) - \text{SNR}(t)) \, ||\mathbf{x} - \hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t)||_2^2 \right],$$

For the specific choices of $\sigma_t, \alpha_t, \hat{\mathbf{x}}_\theta(\mathbf{z}_t; t)$ described previously, we get:

$$\mathcal{L}_T(\mathbf{x}) = \frac{T}{2} \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I}), i\sim U\{1,T\}} \left[(\exp(\gamma_{\boldsymbol{\eta}}(t) - \gamma_{\boldsymbol{\eta}}(s)) - 1) \, \|\boldsymbol{\epsilon} - \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{z}_t; t)\|_2^2 \right]$$

We can jointly optimize γ, η by maximizing the Monte Carlo estimator of this loss.

Note $\exp(\cdot) - 1$ is a common primitive $expm1(\cdot)$ in numerical computing packages can be stably implemented in 32-bit or lower precision, allowing for lower precision implementations of discrete-time diffusion models.

Continuous-time model

It can mathematically be shown that the VLB will be better for larger number of timesteps. This motivates the usage of continuous time where $T \to \infty$

Taking this limit gives the following loss:

$$\begin{aligned}\mathcal{L}_\infty(\mathbf{x}) &= -\frac{1}{2}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})}\int_0^1 \mathrm{SNR}'(t)\,\|\mathbf{x}-\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t;t)\|_2^2\,dt,\\ &= -\frac{1}{2}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I}),t\sim\mathcal{U}(0,1)}\left[\mathrm{SNR}'(t)\,\|\mathbf{x}-\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_t;t)\|_2^2\right].\end{aligned}$$

where $\mathrm{SNR}'(t) = d\mathrm{SNR}(t)/dt$

For the specific choices of $\sigma_t, \alpha_t, \hat{\mathbf{x}}_\theta(\mathbf{z}_t;t)$ described previously, we get:

$$\mathcal{L}_\infty(\mathbf{x}) = \frac{1}{2}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I}),t\sim\mathcal{U}(0,1)}\left[\gamma'_{\boldsymbol{\eta}}(t)\,\|\boldsymbol{\epsilon}-\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{z}_t;t)\|_2^2\right],$$

Once again, optimized and evaluated with Monte Carlo estimator

Reparameterization of the model

Since $\mathrm{SNR}(t)$ is strictly monotonically decreasing in time, it is invertible and we can define v as the SNR at timestep $t = SNR^{-1}(v)$.

We can instead have α_v, σ_v, $\mathbf{z}_v$, and $\tilde{\mathbf{x}}_\theta(\mathbf{z}_v; v)$, and rewrite the loss:

$$\mathcal{L}_\infty(\mathbf{x}) = \frac{1}{2}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})} \int_{\mathrm{SNR_{min}}}^{\mathrm{SNR_{max}}} \|\mathbf{x} - \tilde{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_v, v)\|_2^2 \, dv,$$

where $\mathrm{SNR}_{min} = \mathrm{SNR}(0)$ and $\mathrm{SNR}_{max} = \mathrm{SNR}(1)$

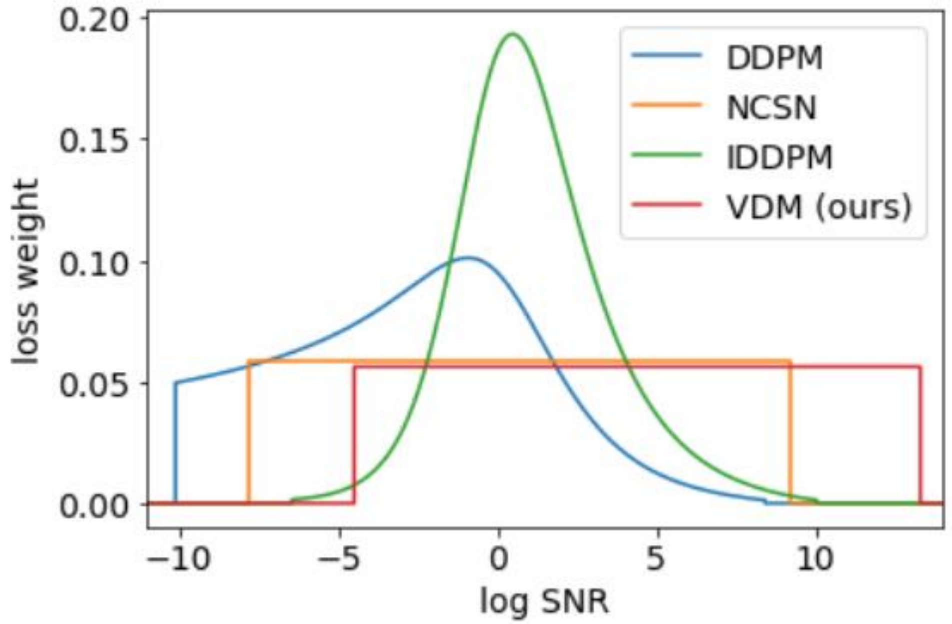
α_t & σ_t only at the endpoints affect the diffusion loss, the loss is invariant to the actual shape of the SNR function

Any SNR function (inc. VP and VE) gives equivalent diffusion models

Weighted diffusion loss

A weighted diffusion loss (which includes previous models like DDPM and NCSN) has similar properties:

$$\mathcal{L}_\infty(\mathbf{x}, w) = \frac{1}{2}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,\mathbf{I})} \int_{\mathrm{SNR_{min}}}^{\mathrm{SNR_{max}}} w(v)\,\|\mathbf{x} - \tilde{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{z}_v, v)\|_2^2\, dv,$$



Results

Continuous time formulation, learned variance, variance minimization, and Fourier features all improve performance as demonstrated by ablation studies

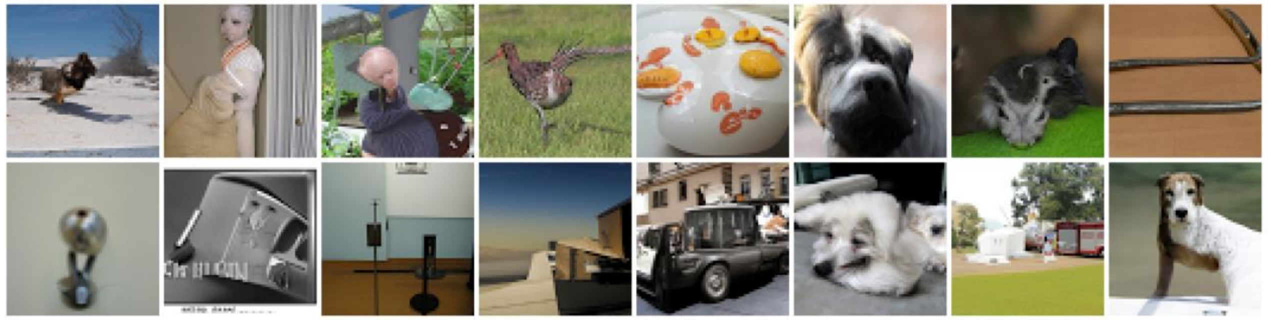
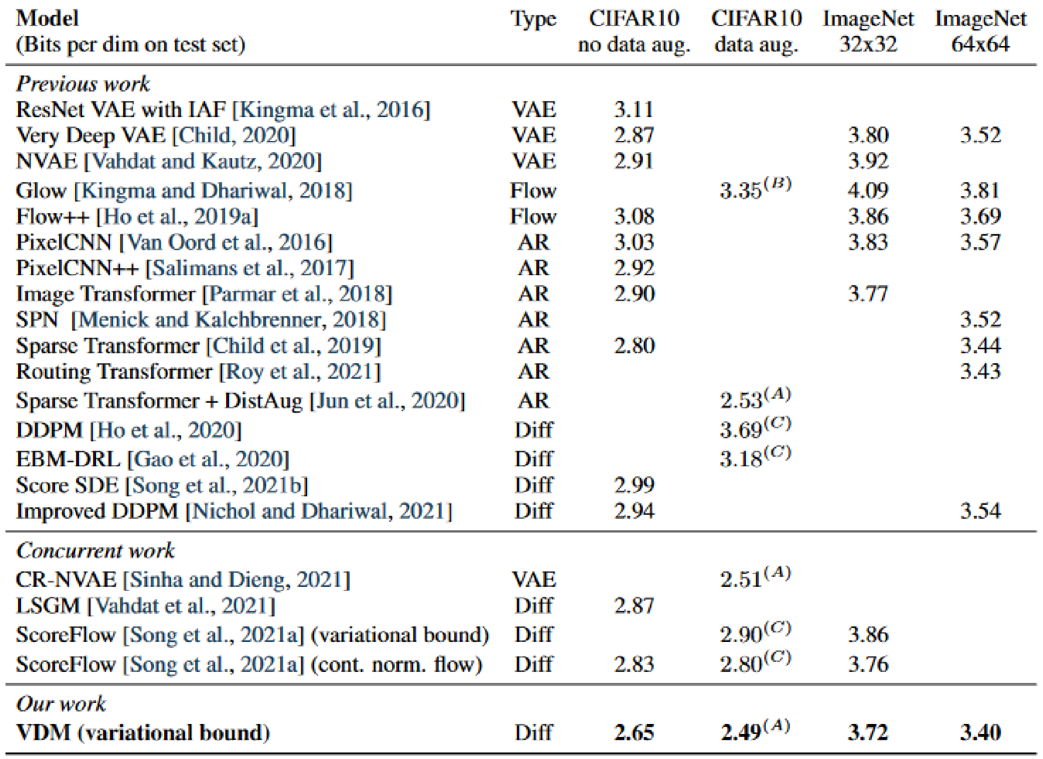


Figure 3: Non cherry-picked unconditional samples from our Imagenet 64x64 model, trained in continuous time and generated using $T = 1000$. The model's hyper-parameters and parameters are optimized w.r.t. the likelihood bound, so the model is not optimized for synthesis quality.



Model (Bits per dim on test set)	Type	CIFAR10 no data aug.	CIFAR10 data aug.	ImageNet 32x32	ImageNet 64x64
Previous work					
ResNet VAE with IAF [Kingma et al., 2016]	VAE	3.11			
Very Deep VAE [Child, 2020]	VAE	2.87		3.80	3.52
NVAE [Vahdat and Kautz, 2020]	VAE	2.91		3.92	
Glow [Kingma and Dhariwal, 2018]	Flow		$3.35^{(B)}$	4.09	3.81
Flow++ [Ho et al., 2019a]	Flow	3.08		3.86	3.69
PixelCNN [Van Oord et al., 2016]	AR	3.03		3.83	3.57
PixelCNN++ [Salimans et al., 2017]	AR	2.92			
Image Transformer [Parmar et al., 2018]	AR	2.90		3.77	
SPN [Menick and Kalchbrenner, 2018]	AR				3.52
Sparse Transformer [Child et al., 2019]	AR	2.80			3.44
Routing Transformer [Roy et al., 2021]	AR				3.43
Sparse Transformer + DistAug [Jun et al., 2020]	AR		$2.53^{(A)}$		
DDPM [Ho et al., 2020]	Diff		$3.69^{(C)}$		
EBM-DRL [Gao et al., 2020]	Diff		$3.18^{(C)}$		
Score SDE [Song et al., 2021b]	Diff	2.99			
Improved DDPM [Nichol and Dhariwal, 2021]	Diff	2.94			3.54
Concurrent work					
CR-NVAE [Sinha and Dieng, 2021]	VAE		$2.51^{(A)}$		
LSGM [Vahdat et al., 2021]	Diff	2.87			
ScoreFlow [Song et al., 2021a] (variational bound)	Diff		$2.90^{(C)}$	3.86	
ScoreFlow [Song et al., 2021a] (cont. norm. flow)	Diff	2.83	$2.80^{(C)}$	3.76	
Our work					
VDM (variational bound)	Diff	**2.65**	**$2.49^{(A)}$**	**3.72**	**3.40**

Tackling the Generative Learning Trilemma with Denoising Diffusion GANs

The Generative Learning Trilemma

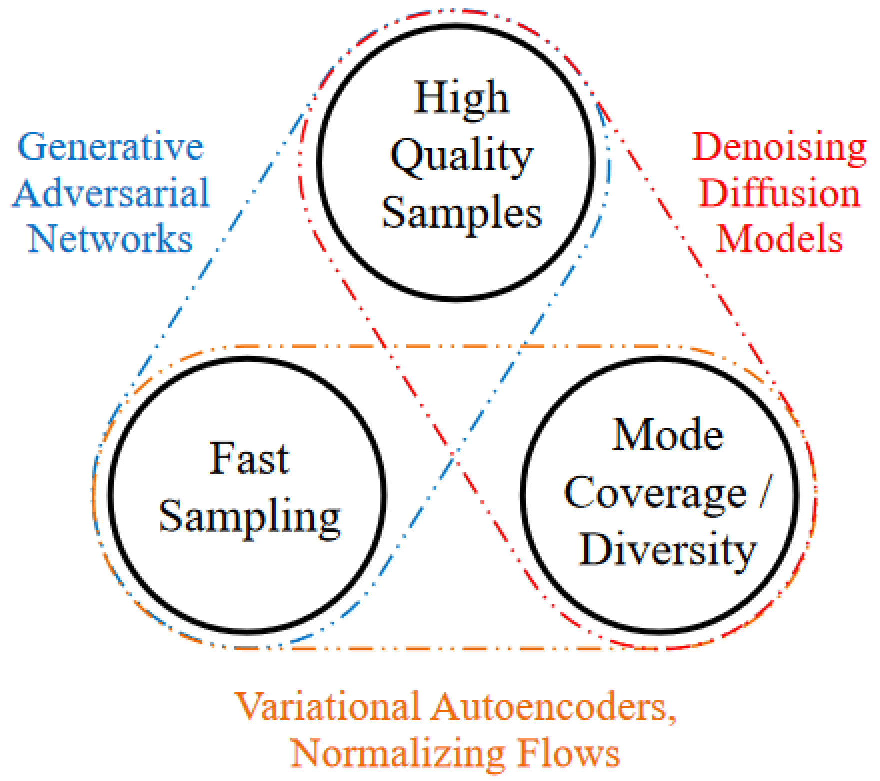
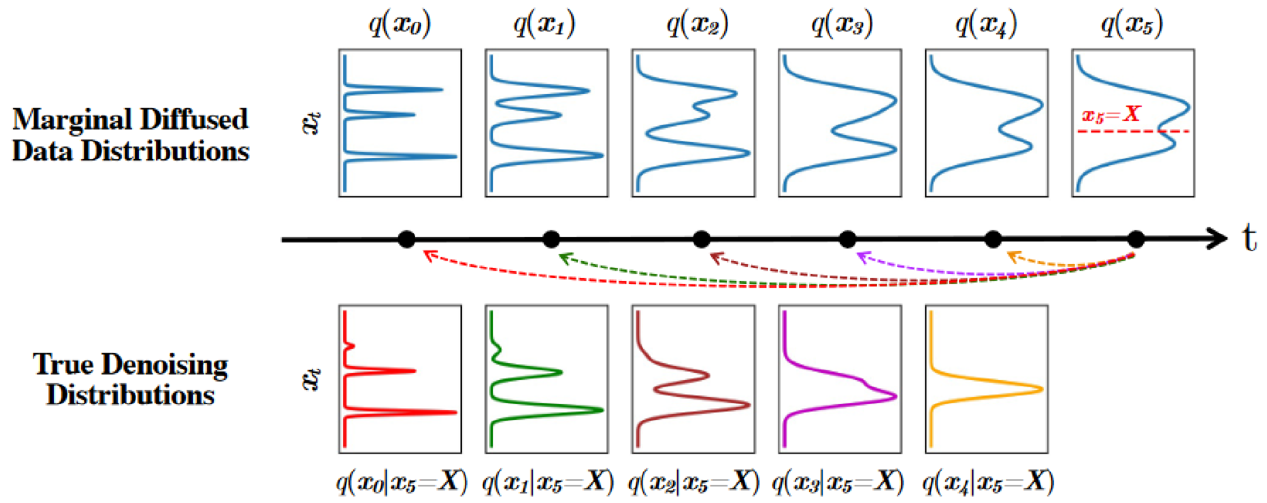

High Quality Samples
Generative Adversarial Networks
Denoising Diffusion Models
Fast Sampling
Mode Coverage / Diversity
Variational Autoencoders, Normalizing Flows

Figure 1: Generative learning trilemma.

Incorrect assumptions under few sampling steps

The reverse (denoising) process $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is assumed to be a Gaussian distribution

With fewer denoising steps, this assumption no longer holds true, and the true denoising distribution is multimodal:



Denoising Diffusion GAN

Introduce a time-dependent discriminator $D_\phi(\mathbf{x}_{t-1}, \mathbf{x}_t, t)$ that decides if $\mathbf{x}_{t-1}$ is a plausible denoised version of $\mathbf{x}_t$

$$\mathbb{E}_{q(\mathbf{x}_t)q(\mathbf{x}_{t-1}|\mathbf{x}_t)}[-\log(D_\phi(\mathbf{x}_{t-1}, \mathbf{x}_t, t))] = \mathbb{E}_{q(\mathbf{x}_0)q(\mathbf{x}_{t-1}|\mathbf{x}_0)q(\mathbf{x}_t|\mathbf{x}_{t-1})}[-\log(D_\phi(\mathbf{x}_{t-1}, \mathbf{x}_t, t))].$$

The generator is also given a random latent vector:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \int p_\theta(\mathbf{x}_0|\mathbf{x}_t)q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)d\mathbf{x}_0 = \int p(\mathbf{z})q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0 = G_\theta(\mathbf{x}_t, \mathbf{z}, t))d\mathbf{z},$$

Denoising Diffusion GAN

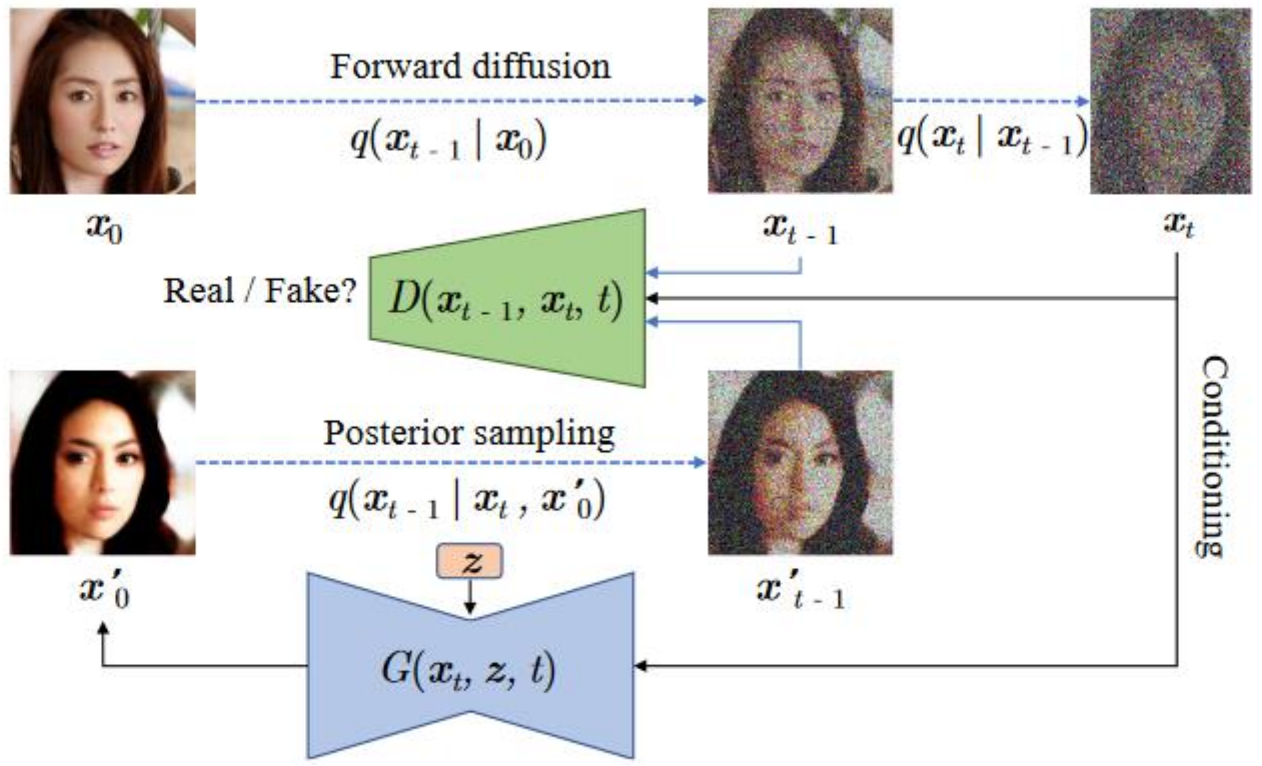


Figure 3: The training process of denoising diffusion GAN.

Advantages over standard GANs

This model breaks the generation process into several conditional denoising diffusion steps in which each step is relatively simple to model, due to the strong conditioning on $\mathbf{x}_t$

Additionally, the diffusion process smoothens the data distribution making the discriminator less likely to overfit.

Empirically, training was stable with no loss explosions

Results

Used a standard U-net, the latent variable controlled adaptive group normalization layers in the network, timestep as embeddings

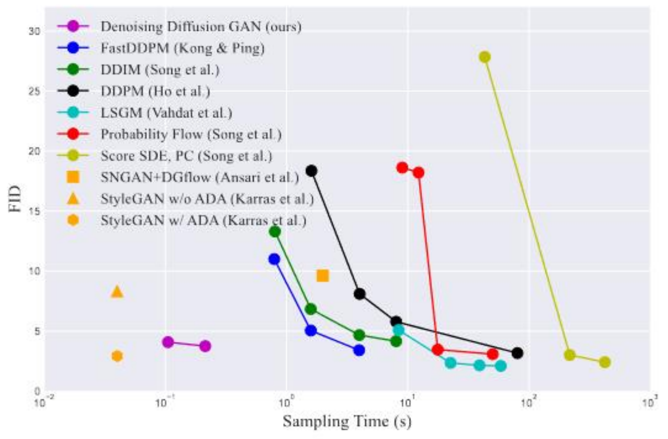


Figure 4: Sample quality vs sampling time trade-off.

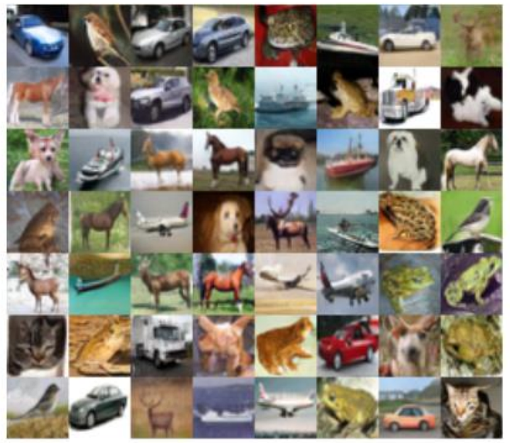
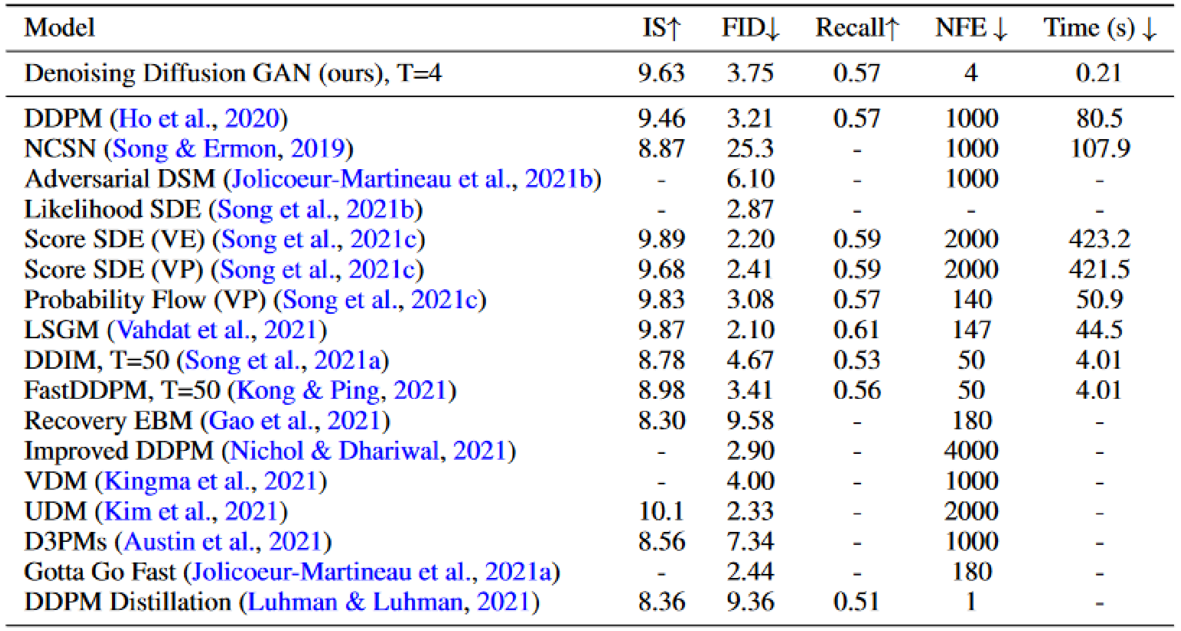


Figure 5: CIFAR-10 qualitative samples.



Model	IS↑	FID↓	Recall↑	NFE ↓	Time (s) ↓
Denoising Diffusion GAN (ours), T=4	9.63	3.75	0.57	4	0.21
DDPM (Ho et al., 2020)	9.46	3.21	0.57	1000	80.5
NCSN (Song & Ermon, 2019)	8.87	25.3	-	1000	107.9
Adversarial DSM (Jolicoeur-Martineau et al., 2021b)	-	6.10	-	1000	-
Likelihood SDE (Song et al., 2021b)	-	2.87	-	-	-
Score SDE (VE) (Song et al., 2021c)	9.89	2.20	0.59	2000	423.2
Score SDE (VP) (Song et al., 2021c)	9.68	2.41	0.59	2000	421.5
Probability Flow (VP) (Song et al., 2021c)	9.83	3.08	0.57	140	50.9
LSGM (Vahdat et al., 2021)	9.87	2.10	0.61	147	44.5
DDIM, T=50 (Song et al., 2021a)	8.78	4.67	0.53	50	4.01
FastDDPM, T=50 (Kong & Ping, 2021)	8.98	3.41	0.56	50	4.01
Recovery EBM (Gao et al., 2021)	8.30	9.58	-	180	-
Improved DDPM (Nichol & Dhariwal, 2021)	-	2.90	-	4000	-
VDM (Kingma et al., 2021)	-	4.00	-	1000	-
UDM (Kim et al., 2021)	10.1	2.33	-	2000	-
D3PMs (Austin et al., 2021)	8.56	7.34	-	1000	-
Gotta Go Fast (Jolicoeur-Martineau et al., 2021a)	-	2.44	-	180	-
DDPM Distillation (Luhman & Luhman, 2021)	8.36	9.36	0.51	1	-

Results

Table 2: Ablation studies on CIFAR-10.

Model Variants	IS↑	FID↓	Recall↑
T = 1	8.93	14.6	0.19
T = 2	**9.80**	4.08	0.54
T = 4	9.63	**3.75**	**0.57**
T = 8	9.43	4.36	0.56
One-shot w/ aug	8.96	13.2	0.25
Direct denoising	9.10	6.03	0.53
Noise generation	8.79	8.04	0.52
No latent variable	8.37	20.6	0.42

Table 3: Mode coverage on StackedMNIST.

Model	Modes↑	KL↓
VEEGAN (Srivastava et al.)	762	2.173
PacGAN (Lin et al.)	992	0.277
PresGAN (Dieng et al.)	**1000**	0.115
InclusiveGAN (Yu et al.)	997	0.200
StyleGAN2 (Karras et al.)	940	0.424
Adv. DSM (Jolicoeur-Martineau et al.)	**1000**	1.49
VAEBM (Xiao et al.)	**1000**	0.087
Denoising Diffusion GAN (ours)	**1000**	**0.071**

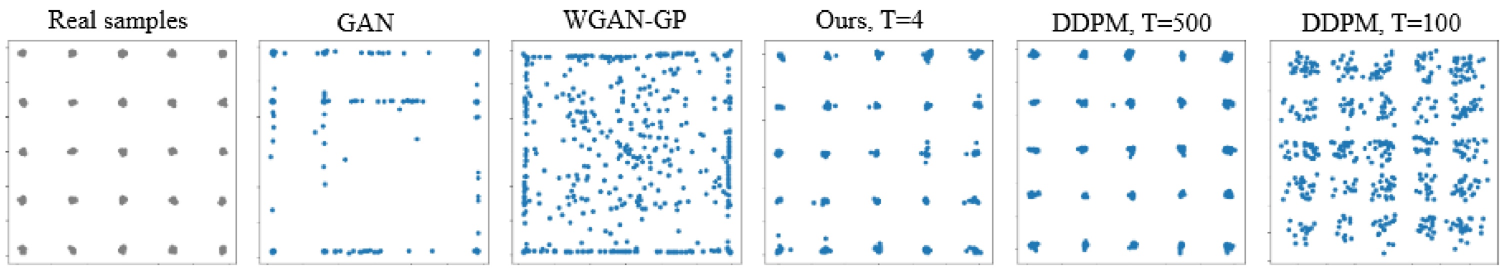


Figure 6: Qualitative results on the 25-Gaussians dataset.