

Diffusion Study Group #6

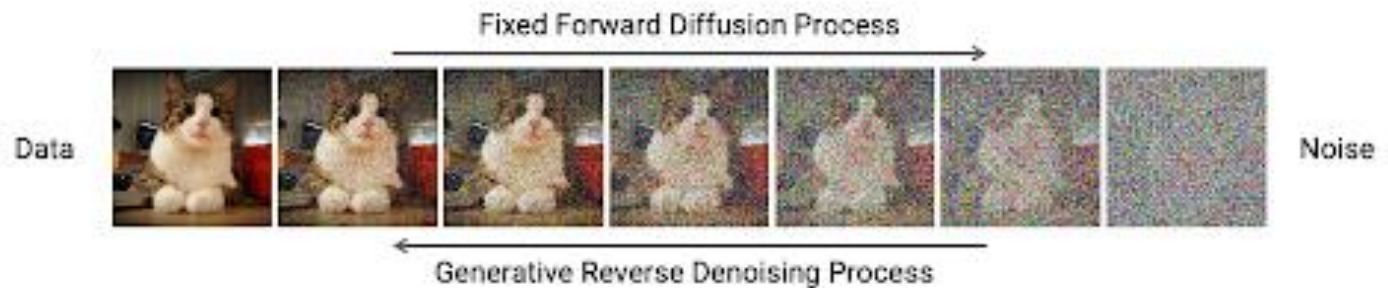
10/22/2022

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!

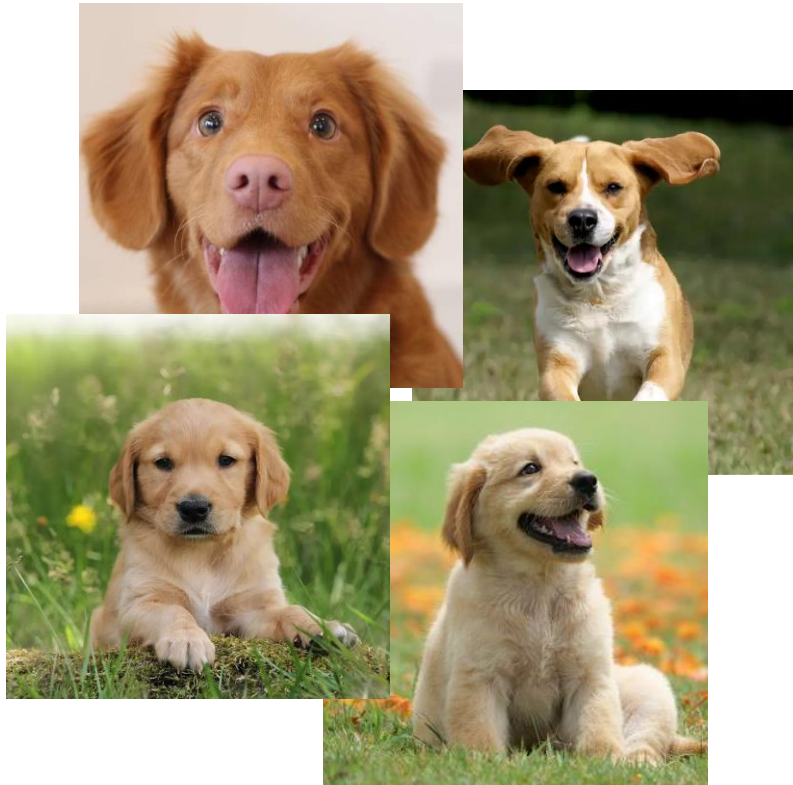


Denoising Diffusion Probabilistic Models

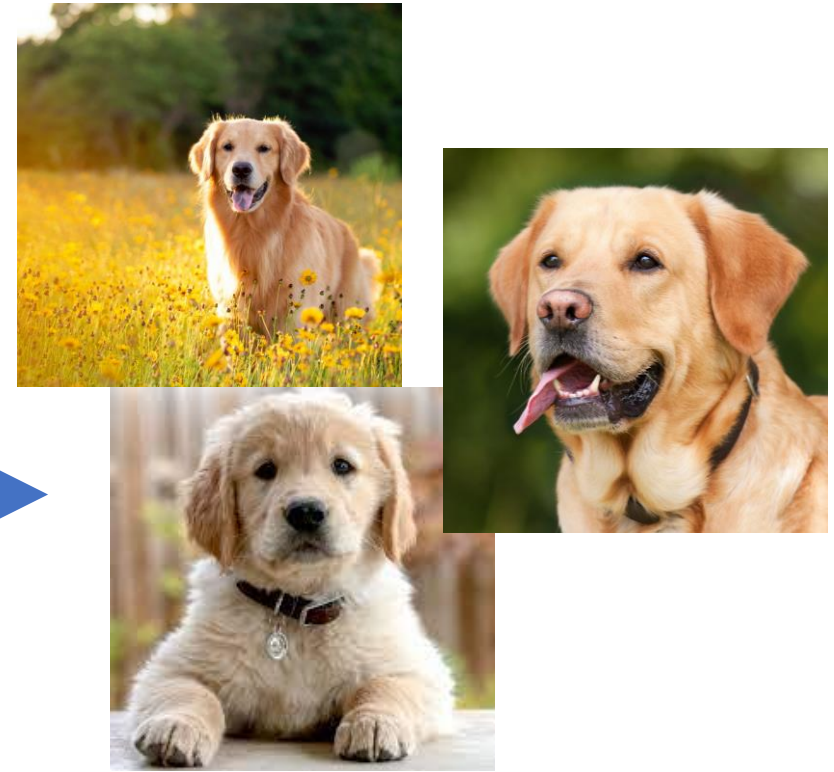
Jonathan Ho, Ajay Jain, Pieter Abbeel

June 19th, 2020

What's the task?



Given these datapoints...



Can we generate more like it?

Forward process

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



\mathbf{x}_{25}



\mathbf{x}_{60}

Observed image



\mathbf{x}_{75}



$\mathbf{x}_{100} = \mathbf{x}_T$

Equivalent to Gaussian noise

Reverse process

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



\mathbf{x}_0

Generated image!



\mathbf{x}_{25}



\mathbf{x}_{50}



\mathbf{x}_{75}



$\mathbf{x}_{100} = \mathbf{x}_T$

Equivalent to Gaussian noise

How to train diffusion models?

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

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

Therefore we must minimize:

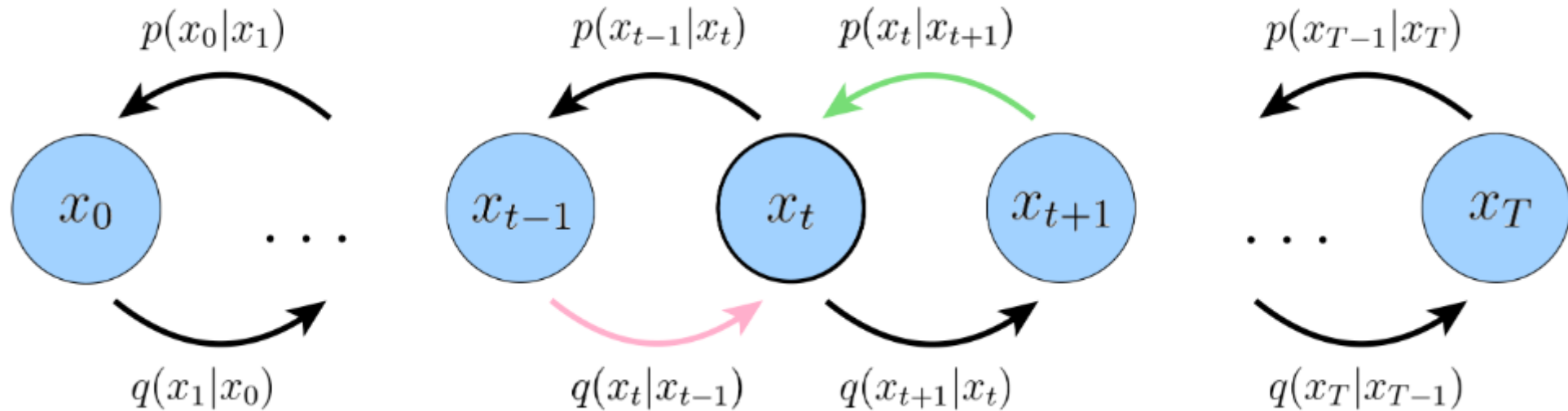
$$\mathcal{L} = E_{\mathbf{x}_0 \sim p_{data}} [-\log p_{\theta}(\mathbf{x}_0)]$$

This requires:

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

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:

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

where:

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

Model sampling

Algorithm 2 Sampling

```
1:  $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
2: for  $t = T, \dots, 1$  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 $\mathbf{x}_t, \boldsymbol{\epsilon}_\theta, t$, get predicted denoised image $\mathbf{x}_0 \approx \hat{\mathbf{x}}_0 = (\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t)) / \sqrt{\bar{\alpha}_t}$
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 \rightarrow score-based model

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

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

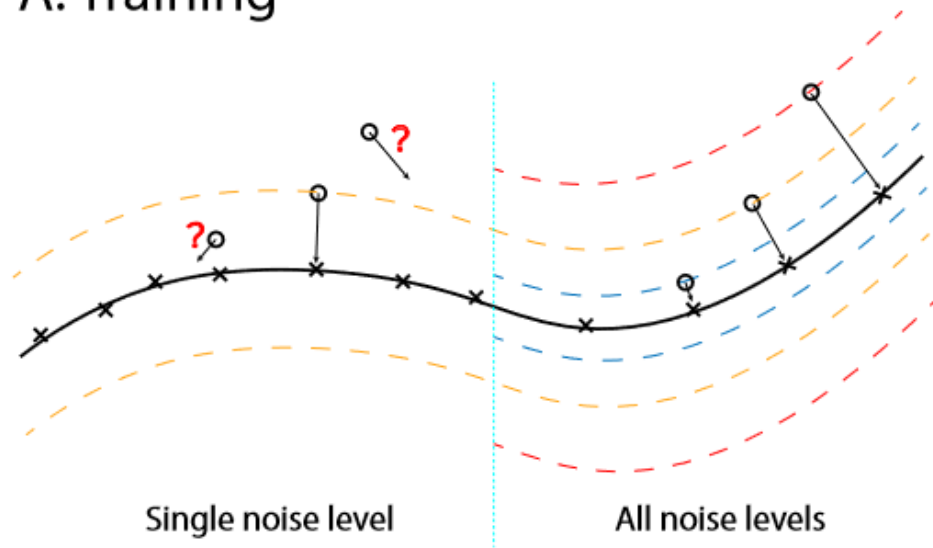
Denoising is equivalent to score matching:

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

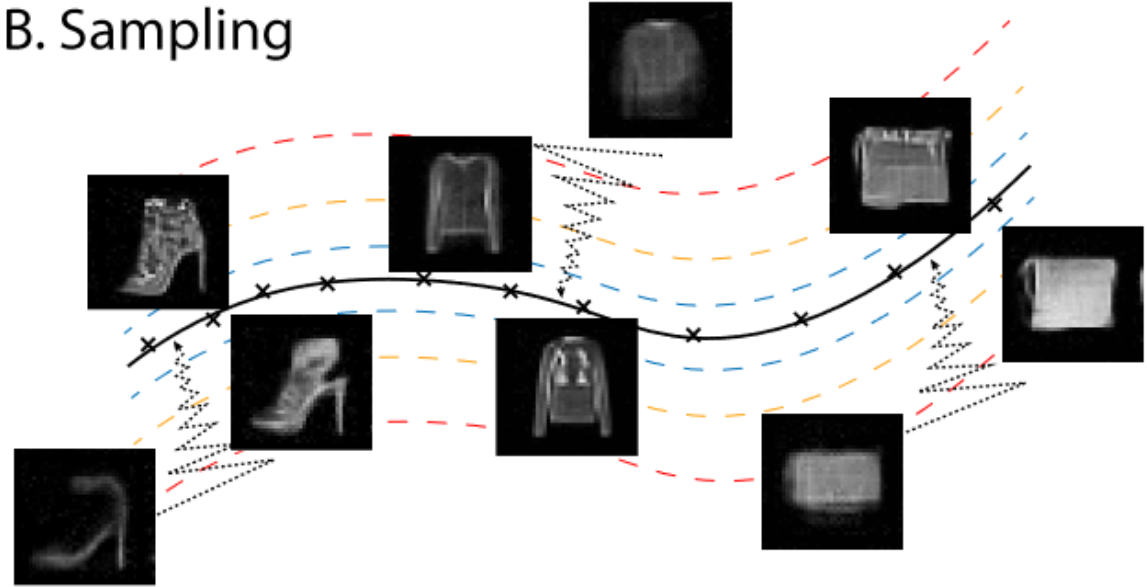
$$\boldsymbol{\epsilon}_{\theta}(\mathbf{x}_t, t) = -\sqrt{1 - \bar{\alpha}_t} \mathbf{s}_{\theta}(\mathbf{x}_t, t)$$

Score matching perspective - Learning the data manifold!

A. Training



B. Sampling



Comparing Noise-conditional Score Networks (NCSN) and DDPM

DDPM noisy distribution:

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

NCSN noisy distribution:

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

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)

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

Solution:

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

Numerical solution:

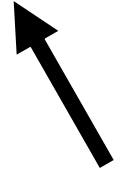
$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t), t)\Delta t$$

Stochastic Differential Equations (SDEs)

Like ODEs... *but with noise!*

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

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 $\Delta t \rightarrow 0$, 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 $p_{st}(\mathbf{x}(t)|\mathbf{x}(s))$

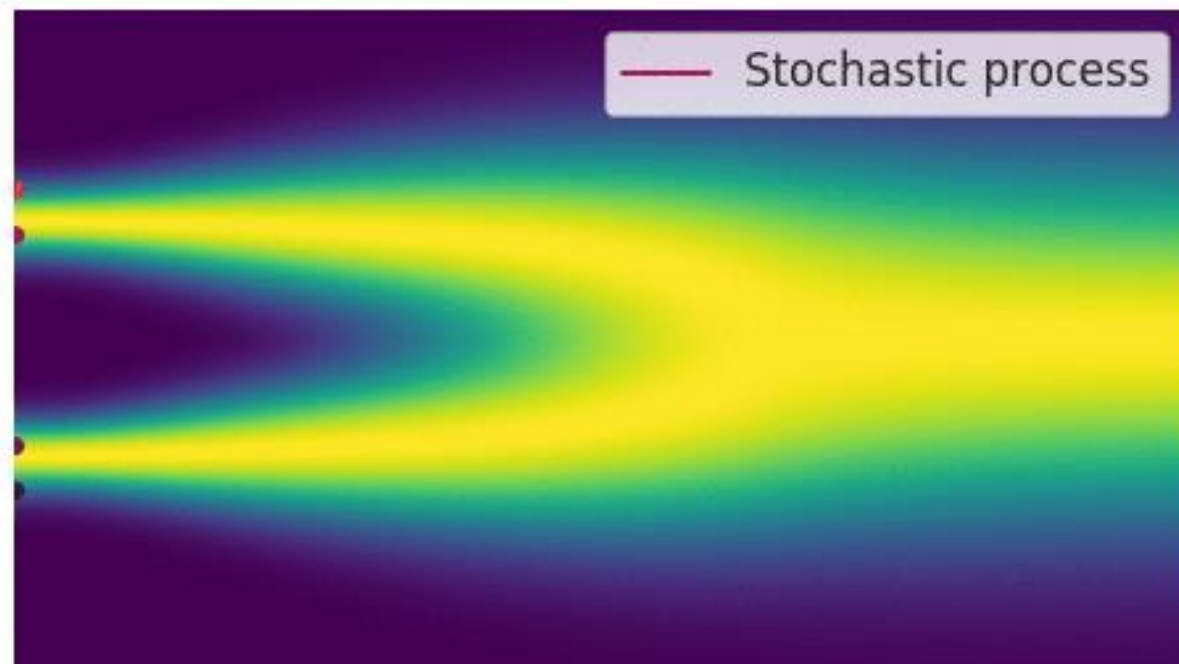
An important property

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

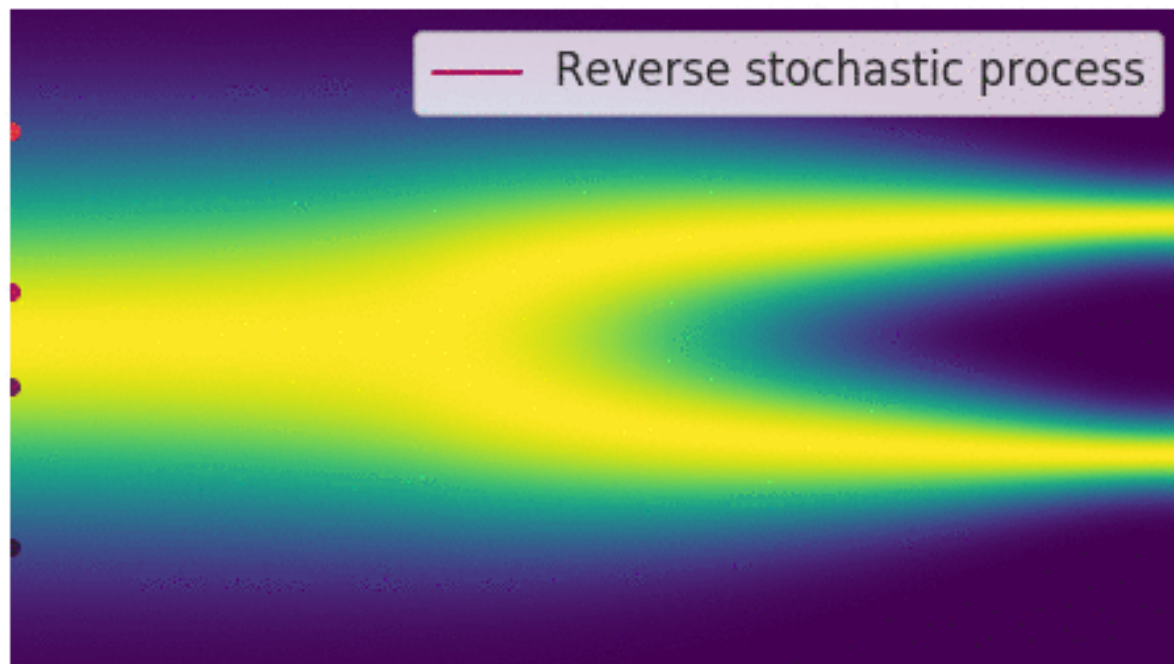
$$d\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - g^2(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x})]dt + g(t)d\bar{\mathbf{w}}$$

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):

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

DDPM (VP SDE):

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

Sub-VP SDE:

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

Training the score function for the reverse SDE

The training objective with denoising score matching is as follows:

$$\theta^* = \arg \min_{\theta} \mathbb{E}_t \left\{ \lambda(t) \mathbb{E}_{\mathbf{x}(0)} \mathbb{E}_{\mathbf{x}(t)|\mathbf{x}(0)} \left[\left\| \mathbf{s}_{\theta}(\mathbf{x}(t), t) - \nabla_{\mathbf{x}(t)} \log p_{0t}(\mathbf{x}(t) | \mathbf{x}(0)) \right\|_2^2 \right] \right\}.$$

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

Solving the Reverse SDE

Given the SDE:

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

We have the reverse-time SDE:

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

Which we discretize as such:

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

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

Probability Flow ODEs

Given the SDE:

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

A corresponding “probability flow ODE” can be found:

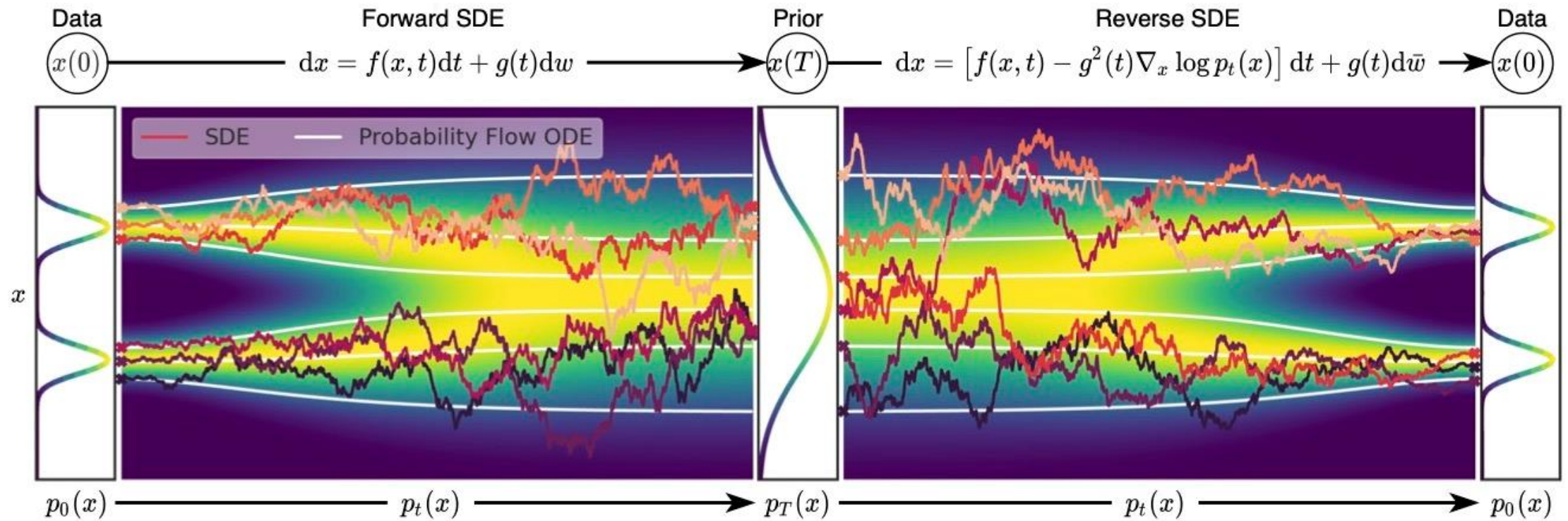
$$d\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - \frac{1}{2} g^2(t) \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) \right] dt$$

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



Uniquely identifiable encodings

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

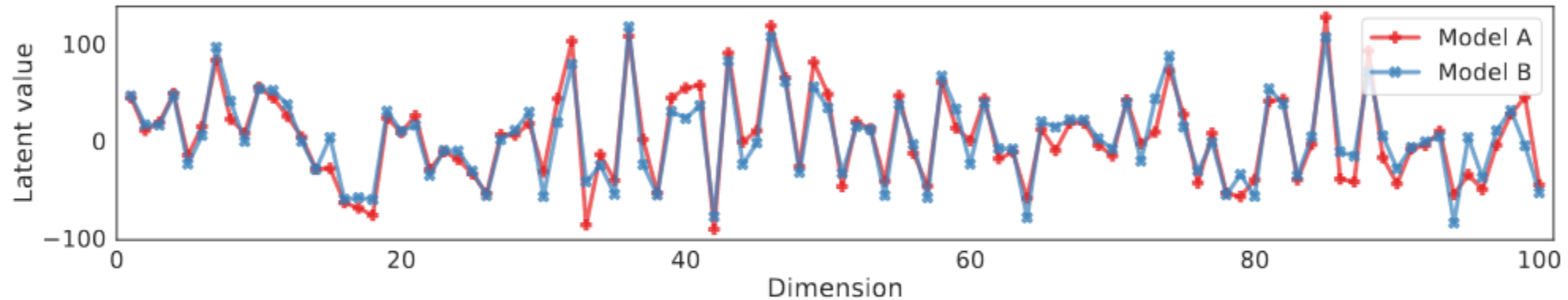
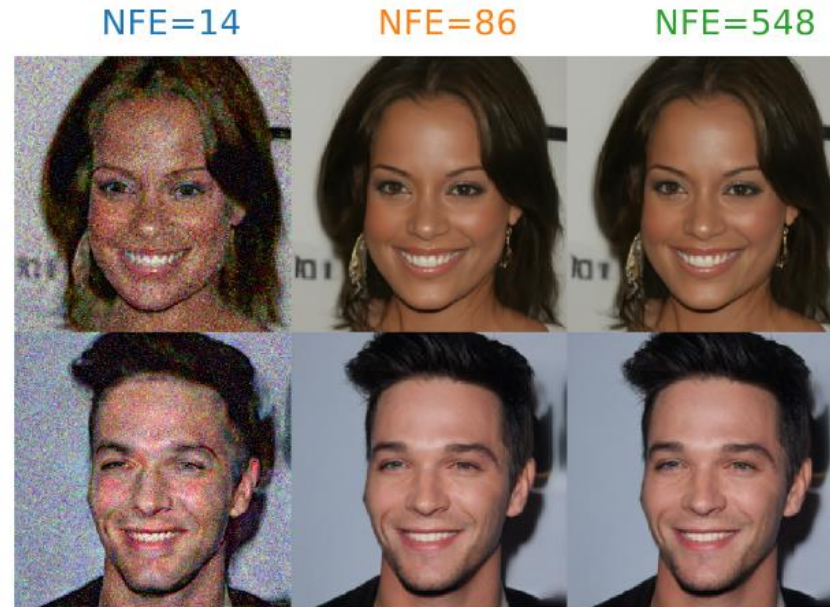
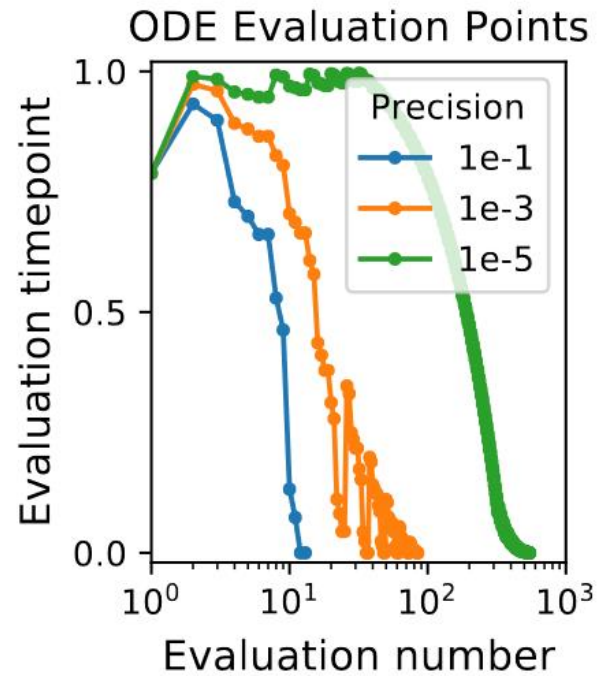


Figure 7: Comparing the first 100 dimensions of the latent code obtained for a random CIFAR-10 image. “Model A” and “Model B” are separately trained with different architectures.

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

$$\begin{aligned} q(\mathbf{x}_t | \mathbf{x}_{t-1}) &= \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t} \mathbf{x}_{t-1}, \beta_t \mathbf{I}) \rightarrow \\ q(\mathbf{x}_t | \mathbf{x}_{t-1}) &= \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t / \alpha_{t-1}} \mathbf{x}_{t-1}, (1 - \alpha_t / \alpha_{t-1}) \mathbf{I}) \end{aligned}$$

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

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

Let us analyze the loss function

Our loss function:

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

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

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 $q_{\sigma}(\mathbf{x}_T \mid \mathbf{x}_0) = \mathcal{N}(\sqrt{\alpha_T}\mathbf{x}_0, (1 - \alpha_T)\mathbf{I})$

Non-Markovian forward processes

The forward process is therefore given by Bayes' rule:

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

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 $\sigma = 0$.

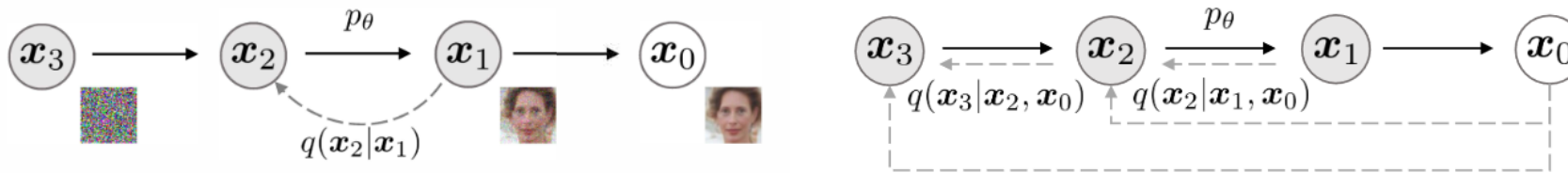


Figure 1: Graphical models for diffusion (left) and non-Markovian (right) inference models.

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:

$$\begin{aligned} J_\sigma(\epsilon_\theta) &:= \mathbb{E}_{\mathbf{x}_{0:T} \sim q_\sigma(\mathbf{x}_{0:T})} [\log q_\sigma(\mathbf{x}_{1:T}|\mathbf{x}_0) - \log p_\theta(\mathbf{x}_{0:T})] \\ &= \mathbb{E}_{\mathbf{x}_{0:T} \sim q_\sigma(\mathbf{x}_{0:T})} \left[\log q_\sigma(\mathbf{x}_T|\mathbf{x}_0) + \sum_{t=2}^T \log q_\sigma(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) - \sum_{t=1}^T \log p_\theta^{(t)}(\mathbf{x}_{t-1}|\mathbf{x}_t) - \log p_\theta(\mathbf{x}_T) \right] \end{aligned} \quad (11)$$

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

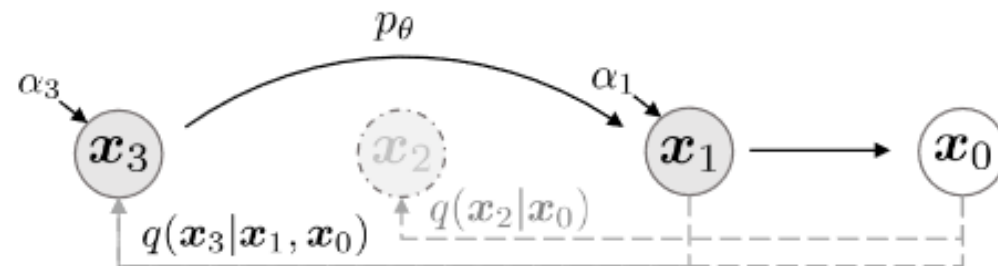
Sampling procedure:

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

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:

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

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

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

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:

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

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: $T = 1000 \rightarrow 4000$, 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:

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

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

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

Where $L_{vlb} = L_0 + L_1 + \dots + L_{T-1} + L_T$ (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 $L_{v\text{lb}}$ to improve training stability (gradient noise scale)

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

Speeding up sampling

Given the original sequence of timesteps $(1, 2, \dots, T)$, 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

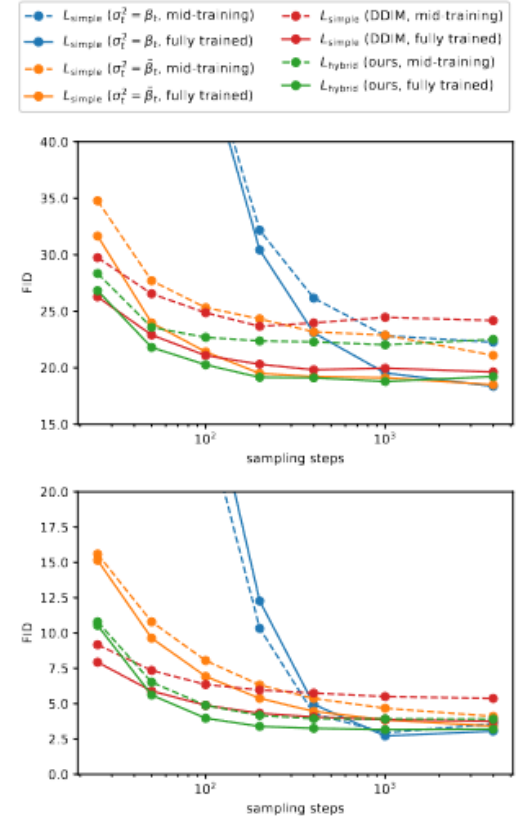


Figure 8. FID versus number of sampling steps, for models trained on ImageNet 64×64 (top) and CIFAR-10 (bottom). All models were trained with 4000 diffusion steps.

Class-conditional DDPM

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

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.



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

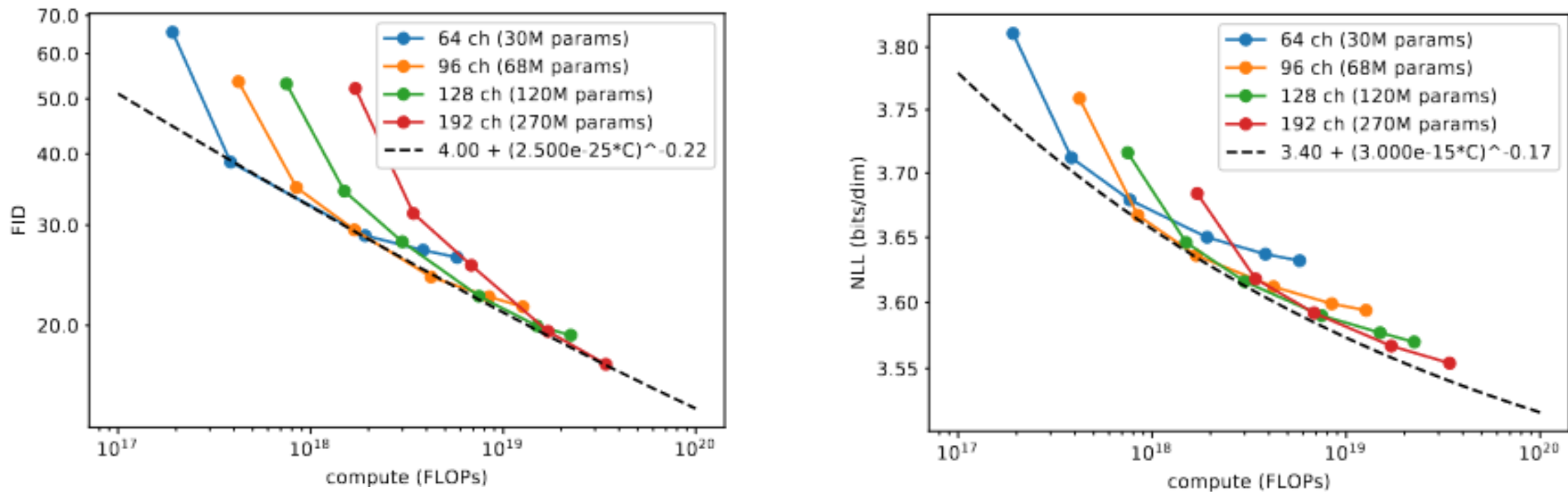


Figure 10. FID and validation NLL throughout training on ImageNet 64×64 for different model sizes. The constant for the FID trend line was approximated using the FID of in-distribution data. For the NLL trend line, the constant was approximated by rounding down the current state-of-the-art NLL (Roy et 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
\times	\times		26.21	6.35	39.70	0.61	0.63
\times	\checkmark	1.0	33.03	6.99	32.92	0.56	0.65
\times	\checkmark	10.0	12.00	10.40	95.41	0.76	0.44
\checkmark	\times		10.94	6.02	100.98	0.69	0.63
\checkmark	\checkmark	1.0	4.59	5.25	186.70	0.82	0.52
\checkmark	\checkmark	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 \times 256 with batch size 256.

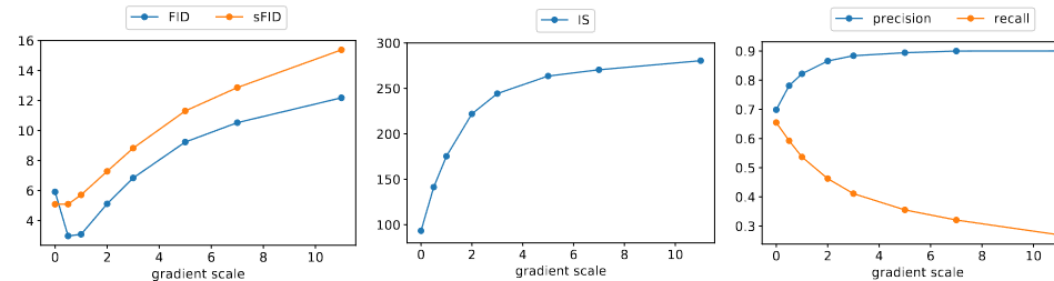


Figure 4: Change in sample quality as we vary scale of the classifier gradients for a class-conditional ImageNet 128 \times 128 model.

Results

Model	FID	sFID	Prec	Rec	Model	FID	sFID	Prec	Rec
LSUN Bedrooms 256×256					ImageNet 128×128				
DCTransformer [†] [42]	6.40	6.66	0.44	0.56	BigGAN-deep [5]	6.02	7.18	0.86	0.35
DDPM [25]	4.89	9.07	0.60	0.45	LOGAN [†] [68]	3.36			
IDDPM [43]	4.24	8.21	0.62	0.46	ADM	5.91	5.09	0.70	0.65
StyleGAN [27]	2.35	6.62	0.59	0.48	ADM-G (25 steps)	5.98	7.04	0.78	0.51
ADM (dropout)	1.90	5.59	0.66	0.51	ADM-G	2.97	5.09	0.78	0.59
LSUN Horses 256×256					ImageNet 256×256				
StyleGAN2 [28]	3.84	6.46	0.63	0.48	DCTransformer [†] [42]	36.51	8.24	0.36	0.67
ADM	2.95	5.94	0.69	0.55	VQ-VAE-2 ^{†‡} [51]	31.11	17.38	0.36	0.57
ADM (dropout)	2.57	6.81	0.71	0.55	IDDPM [‡] [43]	12.26	5.42	0.70	0.62
LSUN Cats 256×256					SR3 ^{†‡} [53]	11.30			
DDPM [25]	17.1	12.4	0.53	0.48	BigGAN-deep [5]	6.95	7.36	0.87	0.28
StyleGAN2 [28]	7.25	6.33	0.58	0.43	ADM	10.94	6.02	0.69	0.63
ADM (dropout)	5.57	6.69	0.63	0.52	ADM-G (25 steps)	5.44	5.32	0.81	0.49
ImageNet 64×64					ADM-G	4.59	5.25	0.82	0.52
BigGAN-deep* [5]	4.06	3.96	0.79	0.48	ImageNet 512×512				
IDDPM [43]	2.92	3.79	0.74	0.62	BigGAN-deep [5]	8.43	8.13	0.88	0.29
ADM	2.61	3.77	0.73	0.63	ADM	23.24	10.19	0.73	0.60
ADM (dropout)	2.07	4.29	0.74	0.63	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

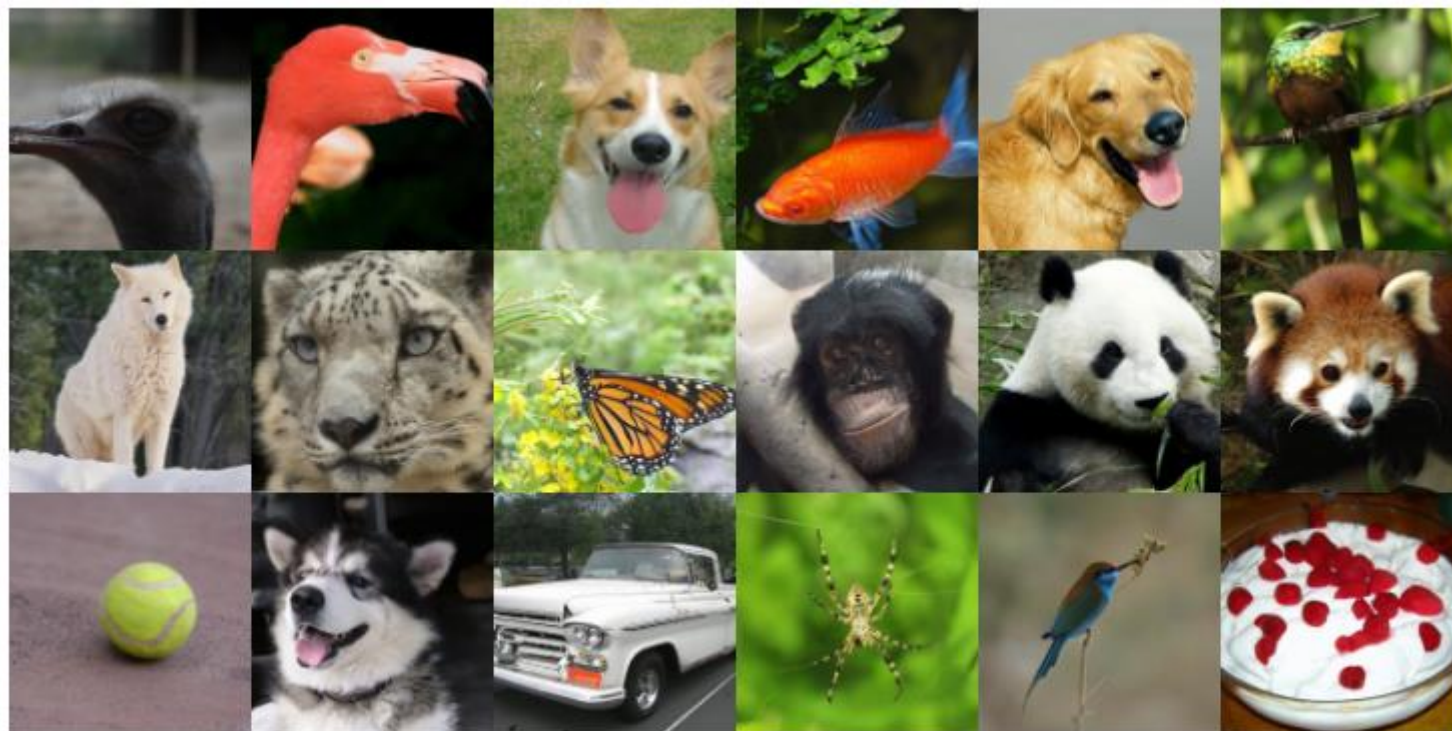


Figure 1: Selected samples from our best ImageNet 512×512 model (FID 3.85)

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