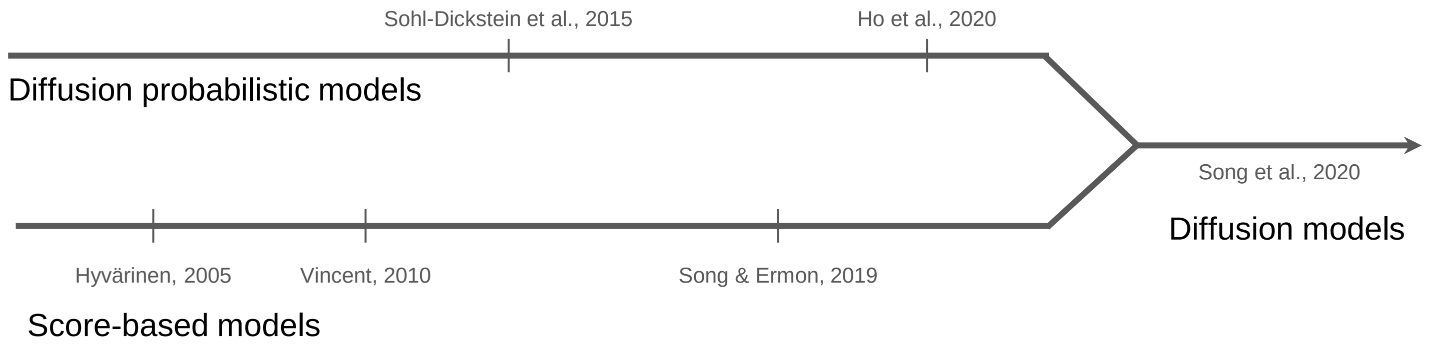
A Unified framework for diffusion models

Sangyun Lee, https://sangyun884.github.io/about/

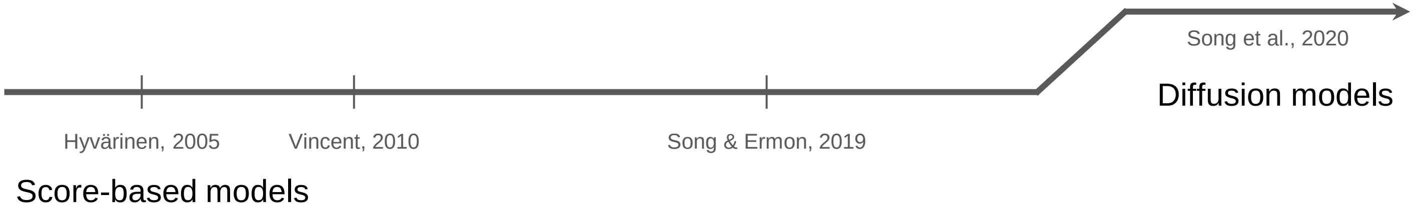
Introduction



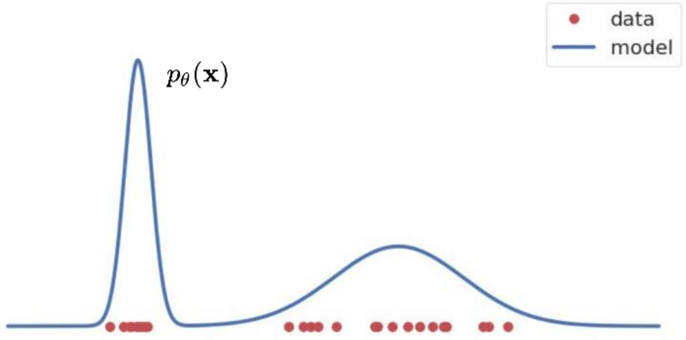
DPMs: natural connection with variational approaches
SBMs: closely related to EBMs (learning an unnormalized density)

Both perspectives are useful for understanding the recently rising iterative methods

Introduction



Generative modeling



$$
\begin{aligned}
\ell(\boldsymbol{\theta}) &= \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x})] \\
&= -D_{\mathrm{KL}}\left(p_{\text{data}}(\mathbf{x}) \parallel p_{\boldsymbol{\theta}}(\mathbf{x})\right) + \text{const}
\end{aligned}
$$

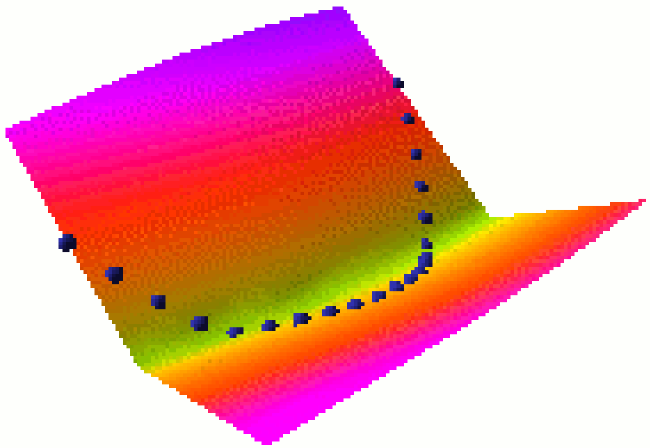
How to construct a flexible pdf?

- Invertible flows – lack of flexibility
- VAEs – surrogate loss
- AR models – sampling speed
- GANs (learning implicit models with min-max game) – training instability

Energy-based models

Gibbs distribution $p_{\boldsymbol{\theta}}(\mathbf{x}) = \dfrac{\exp(-\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}))}{Z_{\boldsymbol{\theta}}}$

Partition function $Z_{\boldsymbol{\theta}} = \displaystyle\int \exp(-\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}))\, \mathrm{d}\mathbf{x}$



Maximum likelihood training

$$\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}) = -\nabla_{\boldsymbol{\theta}}\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}) - \underline{\nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}}}.$$

Difficult to estimate

Energy-based models

$$
\begin{aligned}
\nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}} &= \nabla_{\boldsymbol{\theta}} \log \int \exp(-\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x})) d\mathbf{x} \\
&\overset{(i)}{=} \left(\int \exp(-\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x})) d\mathbf{x} \right)^{-1} \nabla_{\boldsymbol{\theta}} \int \exp(-\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x})) d\mathbf{x} \\
&= \left(\int \exp(-\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x})) d\mathbf{x} \right)^{-1} \int \nabla_{\boldsymbol{\theta}} \exp(-\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x})) d\mathbf{x} \\
&\overset{(ii)}{=} \left(\int \exp(-\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x})) d\mathbf{x} \right)^{-1} \int \exp(-\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}))(-\nabla_{\boldsymbol{\theta}} \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x})) d\mathbf{x} \\
&= \int \left(\int \exp(-\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x})) d\mathbf{x} \right)^{-1} \exp(-\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}))(-\nabla_{\boldsymbol{\theta}} \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x})) d\mathbf{x} \\
&\overset{(iii)}{=} \int \frac{1}{Z_{\boldsymbol{\theta}}} \exp(-\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}))(-\nabla_{\boldsymbol{\theta}} \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x})) d\mathbf{x} \\
&\overset{(iv)}{=} \int p_{\boldsymbol{\theta}}(\mathbf{x})(-\nabla_{\boldsymbol{\theta}} \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x})) d\mathbf{x} \\
&= \underline{\mathbb{E}_{\mathbf{x} \sim p_{\boldsymbol{\theta}}(\mathbf{x})} \left[-\nabla_{\boldsymbol{\theta}} \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}) \right]},
\end{aligned}
$$

To train EBM, we need
to sample from it

Score matching

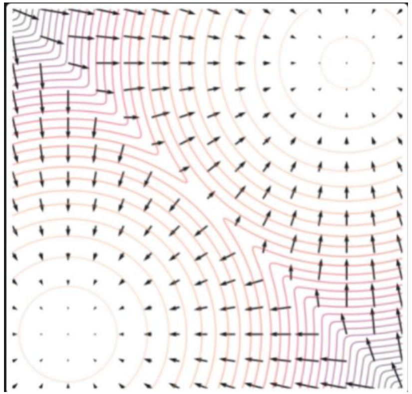
$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \frac{\exp(-\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}))}{Z_{\boldsymbol{\theta}}}$$

$$Z_{\boldsymbol{\theta}} = \int \exp(-\mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x}))\, \mathrm{d}\mathbf{x}$$

Due to the intractable partition function, maximum likelihood training is slow.

Instead, parameterize the log derivative of the density (i.e., score function)
-> the partition function disappears as it is not a function of data

$$\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}) \triangleq \nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x}) = -\nabla_{\mathbf{x}} \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{x})$$



Then, generative modeling is turned into a simple regression task

$$\mathbb{E}_{p_{\mathrm{data}}(\mathbf{x})}\left[\frac{1}{2}\left\|\nabla_{\mathbf{x}} \log p_{\mathrm{data}}(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x})\right\|^2\right].$$

$$= \mathbb{E}_{p_{\mathrm{data}}(\mathbf{x})}\left[\frac{1}{2}\|\boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x})\|^2 + \mathrm{tr}(\mathbf{J}_{\mathbf{x}} \boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x}))\right] + \text{constant}$$

Yet, Jacobian trace of the score network is expensive to compute

Denoising score matching

Score matching in the noised data distribution is easier.

$$q(\tilde{\boldsymbol{x}}|\boldsymbol{x}):\ \text{noise distribution},$$

$$q(\tilde{\boldsymbol{x}}) = \int q(\tilde{\boldsymbol{x}}|\boldsymbol{x}) p_{\text{data}}(\boldsymbol{x}) d\boldsymbol{x}$$

$$\mathbb{E}_{q(\tilde{\mathbf{x}})}\left[\frac{1}{2}\|\nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}) - \nabla_{\mathbf{x}} \log q(\tilde{\mathbf{x}})\|_2^2\right] = \mathbb{E}_{q(\mathbf{x},\tilde{\mathbf{x}})}\left[\frac{1}{2}\|\nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}) - \nabla_{\mathbf{x}} \log q(\tilde{\mathbf{x}}|\mathbf{x})\|_2^2\right] + \text{constant}$$

For a Gaussian perturbation kernel $\nabla_{\mathbf{x}} \log q(\tilde{\mathbf{x}}|\mathbf{x}) = \nabla_{\mathbf{x}} \log \mathcal{N}(\tilde{\mathbf{x}}|\mathbf{x}, \sigma^2\mathbf{I}) = \frac{(\tilde{\mathbf{x}} - \boldsymbol{x})}{\sigma^2}$

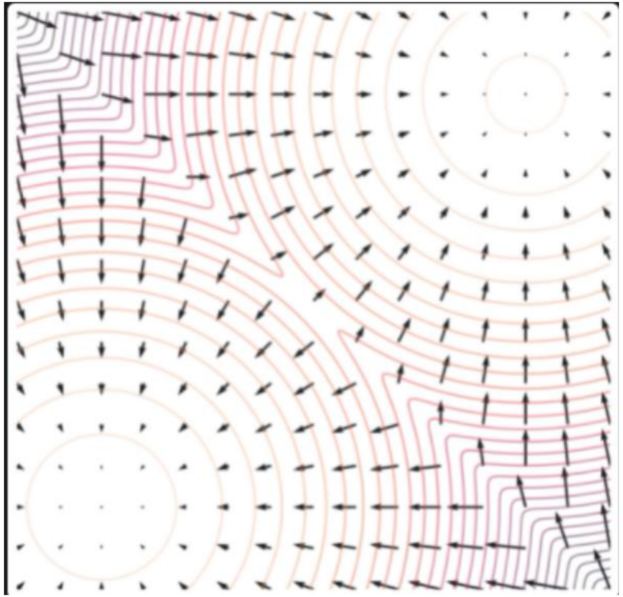
The above objective is reduced to:

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

, which is equivalent to the objective of denoising autoencoders.

However, we cannot learn the score of the true data distribution.

Sampling with Langevin MCMC



Given a score function, we can sample using Langevin MCMC:

$$\tilde{\mathbf{x}}_t = \tilde{\mathbf{x}}_{t-1} + \frac{\epsilon}{2}\nabla_{\mathbf{x}} \log p(\tilde{\mathbf{x}}_{t-1}) + \sqrt{\epsilon}\, \mathbf{z}_t,$$

Gradient ascent   Noise injection

But there are pitfalls…

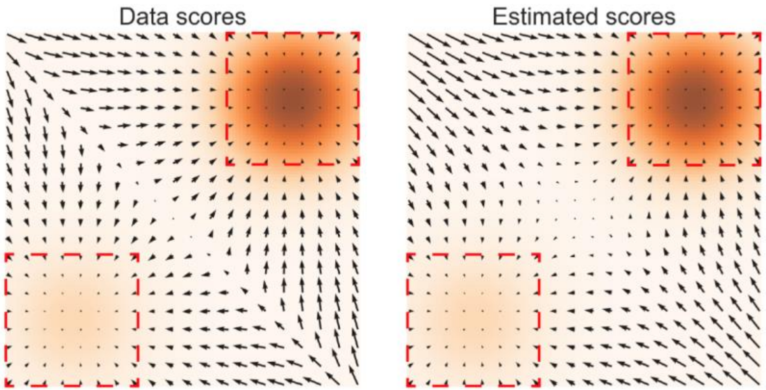
Pitfall 1: Inaccurate score estimation in the low-density regions

The manifold hypothesis tells us that data lies in the low-dimensional manifold.

$$\mathbb{E}_{p_{\text{data}}(\mathbf{x})}\left[\frac{1}{2}\left\|\nabla_{\mathbf{x}}\log p_{\text{data}}(\mathbf{x})-\nabla_{\mathbf{x}}\log p_{\boldsymbol{\theta}}(\mathbf{x})\right\|^2\right].$$

As we sample from data distribution,
the learned score is accurate only at the data manifold.

This is problematic as we usually initialize the MCMC with noises.



Pitfall 2: Relative weights

- Mixture of two disjoint components

$$p_{\text{data}}(\mathbf{x}) = \pi p_1(\mathbf{x}) + (1-\pi)p_2(\mathbf{x})$$

$$\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) = \nabla_{\mathbf{x}} \log p_1(\mathbf{x})$$

No π

$$\pi p_1(\mathbf{x})$$

$$(1-\pi)p_2(\mathbf{x})$$

$$\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) = \nabla_{\mathbf{x}} \log p_2(\mathbf{x})$$

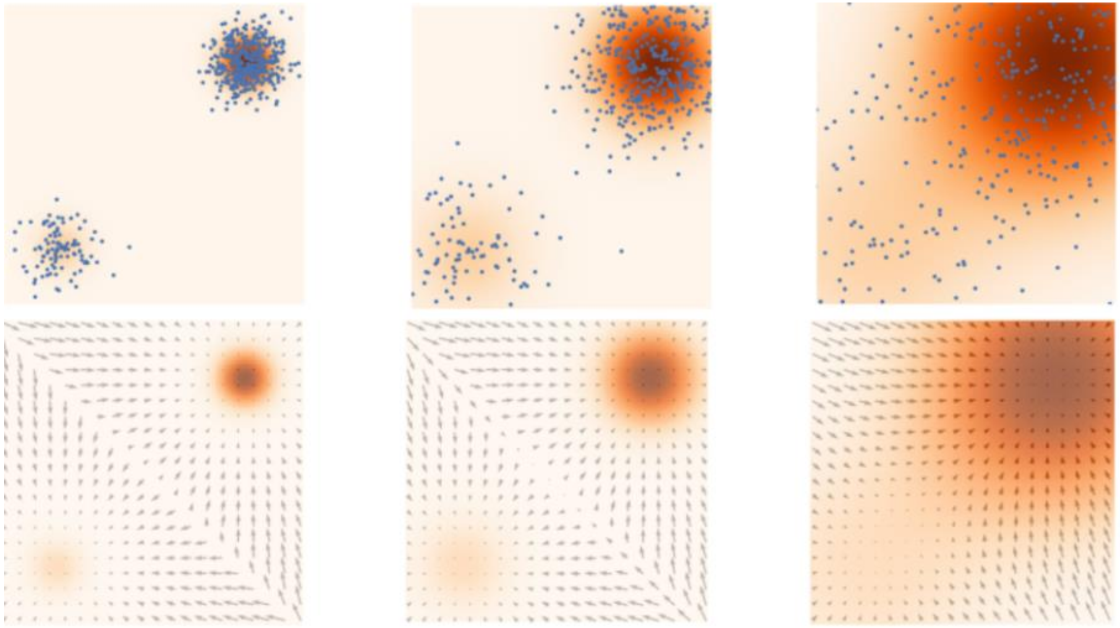
No π

Generative Modeling by Estimating Gradients of the Data Distribution

Yang Song
Stanford University
yangsong@cs.stanford.edu

Stefano Ermon
Stanford University
ermon@cs.stanford.edu

Learning multi-scale score network



Adding Gaussian noise to data can be viewed as “blurring” the data distribution.
But how to determine the noise strength? -> gradually decrease the variance.

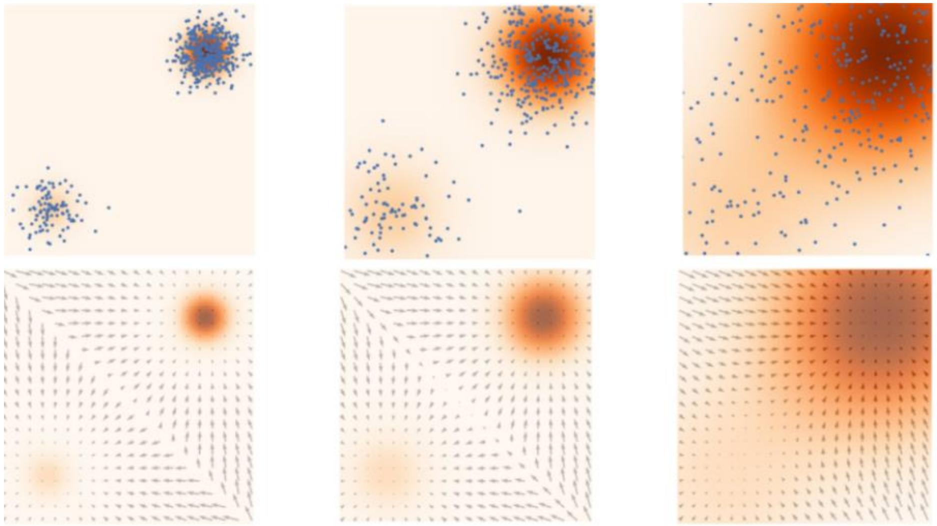
$$\ell(\boldsymbol{\theta};\sigma) \triangleq \frac{1}{2}\mathbb{E}_{p_{\text{data}}(\mathbf{x})}\mathbb{E}_{\tilde{\mathbf{x}}\sim\mathcal{N}(\mathbf{x},\sigma^2 I)}\left[\left\|\mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}},\sigma)+\frac{\tilde{\mathbf{x}}-\mathbf{x}}{\sigma^2}\right\|_2^2\right]. \quad \mathcal{L}(\boldsymbol{\theta};\{\sigma_i\}_{i=1}^L) \triangleq \frac{1}{L}\sum_{i=1}^{L}\lambda(\sigma_i)\ell(\boldsymbol{\theta};\sigma_i),$$

Learning multi-scale score network

Algorithm 1 Annealed Langevin dynamics.

Require: $\{\sigma_i\}_{i=1}^{L}, \epsilon, T$.

Initialize $\tilde{\mathbf{x}}_0$
for $i \leftarrow 1$ to L **do**
  $\alpha_i \leftarrow \epsilon \cdot \sigma_i^2 / \sigma_L^2$ ▷ α_i is the step size.
  for $t \leftarrow 1$ to T **do**
    Draw $\mathbf{z}_t \sim \mathcal{N}(0, I)$
    $\tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \frac{\alpha_i}{2} \mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}_{t-1}, \sigma_i) + \sqrt{\alpha_i}\, \mathbf{z}_t$
  end for
  $\tilde{\mathbf{x}}_0 \leftarrow \tilde{\mathbf{x}}_T$
end for
return $\tilde{\mathbf{x}}_T$



Get a sample from q_{σ_1} $\rightarrow$ Get a sample from q_{σ_2} $\rightarrow$ ... $\rightarrow$ Get a sample from q_{σ_T}, where $\sigma_1 > \sigma_2 > ... > \sigma_T$. σ_T is sufficiently small, so $p_{\sigma_T} \approx p_{data}$.

Learning multi-scale score network

Model	Inception	FID
CIFAR-10 Unconditional		
PixelCNN [59]	4.60	65.93
PixelIQN [42]	5.29	49.46
EBM [12]	6.02	40.58
WGAN-GP [18]	$7.86 \pm .07$	36.4
MoLM [45]	$7.90 \pm .10$	**18.9**
SNGAN [36]	$8.22 \pm .05$	21.7
ProgressiveGAN [25]	$8.80 \pm .05$	-
NCSN (Ours)	$\mathbf{8.87} \pm .12$	25.32
CIFAR-10 Conditional		
EBM [12]	8.30	37.9
SNGAN [36]	$8.60 \pm .08$	25.5
BigGAN [6]	**9.22**	**14.73**

Table 1: Inception and FID scores for CIFAR-10

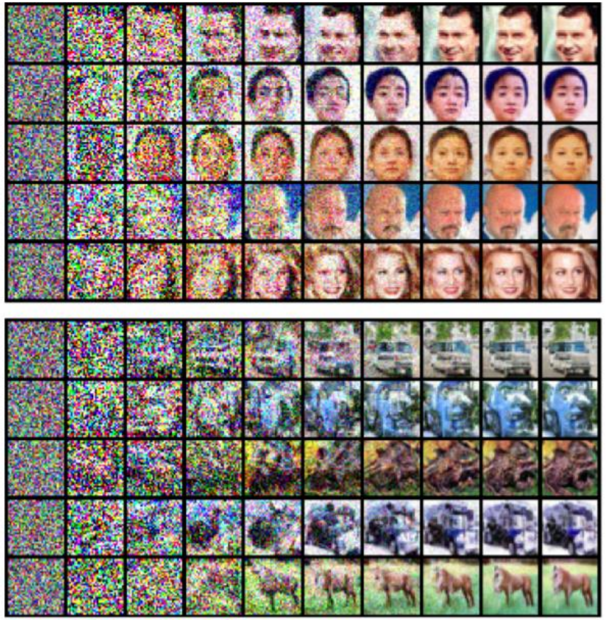


Figure 4: Intermediate samples of annealed Langevin dynamics.

Song & Ermon, Generative Modeling by Estimating Gradients of the Data Distribution, NeurIPS 2019 (Oral)

Connection with stochastic differential equations

SCORE-BASED GENERATIVE MODELING THROUGH STOCHASTIC DIFFERENTIAL EQUATIONS

Yang Song*
Stanford University
yangsong@cs.stanford.edu

Jascha Sohl-Dickstein
Google Brain
jaschasd@google.com

Diederik P. Kingma
Google Brain
durk@google.com

Abhishek Kumar
Google Brain
abhishk@google.com

Stefano Ermon
Stanford University
ermon@cs.stanford.edu

Ben Poole
Google Brain
pooleb@google.com

- A generalized framework that unifies the DPMs and SBMs.
- Reveals connection with continuous-time normalizing flows.
- Controllable generation thanks to the modularity of the score networks.

Stochastic differential equations

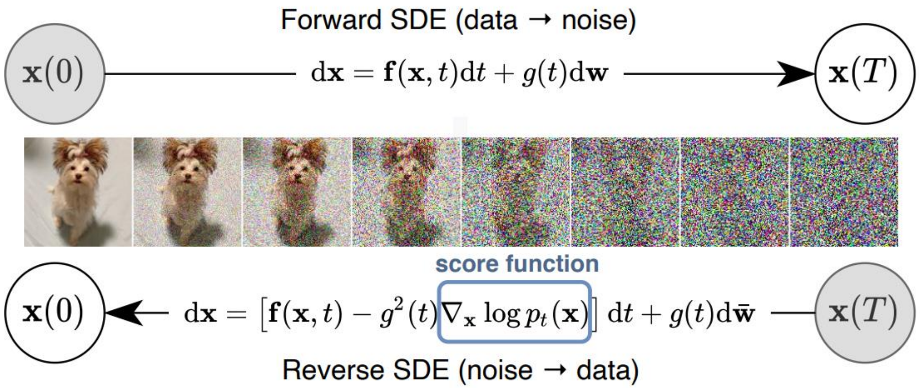
$$d\mathbf{x} = \underbrace{f(\mathbf{x}, t)}_{\text{drift}} dt + \underbrace{g(t)}_{\text{diffusion}} d\mathbf{w}$$

Anderson (1982): a reverse of the diffusion process is also a diffusion process, running backwards in time and given by the reverse-time SDE:

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

Can be estimated by learning
time conditional score network

DPMs and SBMs are the SDEs



Recap: DPMs objective

$$L_{\text{simple}}(\theta) := \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2\right]$$

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

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

ϵ is the score multiplied by a constant -> Unified framework!

DPMs and SBMs are the SDEs

SBMs are the discretized VE-SDEs

$$\mathbf{x}_i = \mathbf{x}_{i-1} + \sqrt{\sigma_i^2 - \sigma_{i-1}^2}\mathbf{z}_{i-1}, \quad i = 1, \cdots, N,$$

-> $$\mathrm{d}\mathbf{x} = \sqrt{\frac{\mathrm{d}\left[\sigma^2(t)\right]}{\mathrm{d}t}}\mathrm{d}\mathbf{w}.$$

DPMs are the discretized VP-SDEs

$$\mathbf{x}_i = \sqrt{1-\beta_i}\mathbf{x}_{i-1} + \sqrt{\beta_i}\mathbf{z}_{i-1}, \quad i = 1, \cdots, N.$$

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

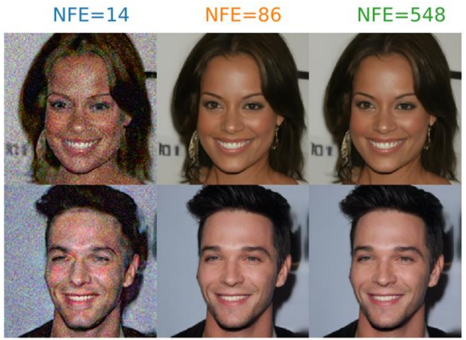
Probability flow ODE

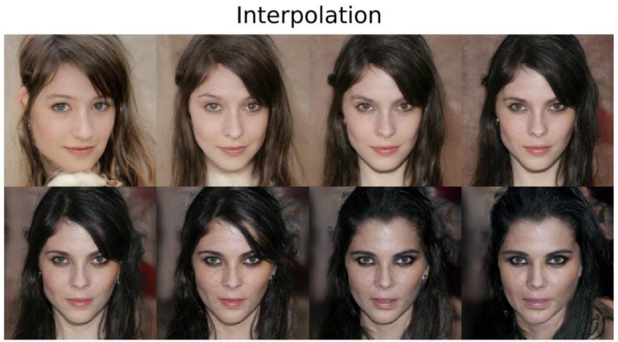
Generative SDE $$d\mathbf{x} = [f(\mathbf{x}, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})]dt + g(t)d\bar{\mathbf{w}}$$

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

PODE yields the same marginal distributions as SDE.

It can be also regarded as a continuous-time flow -> exact likelihood computation!





Fast sampling,
smooth interpolation.

Controllable generation

Inverse problems (from a probabilistic perspective)

$\mathbf{x}$: latent image

$\mathbf{y}$: measurement

Forward model (e.g., adding noise, blurring, downsampling)

$p(\mathbf{y}|\mathbf{x})$

$\mathbf{x}$ $\mathbf{y}$

$p(\mathbf{x}|\mathbf{y})$

Posterior

$$p(\mathbf{x}|\mathbf{y}) = p(\mathbf{y}|\mathbf{x})p(\mathbf{x})/p(\mathbf{y})$$

$$\nabla_{\mathbf{x}} \log p(\mathbf{x}|\mathbf{y}) = \underbrace{\nabla_{\mathbf{x}} \log p(\mathbf{x})}_{\text{score function}} + \nabla_{\mathbf{x}} \log p(\mathbf{y}|\mathbf{x})$$

Controllable generation

Conditional synthesis using unconditional model!

Classifier guidance

Diffusion Models Beat GANs on Image Synthesis

$$x_{t-1} \leftarrow \text{sample from } \mathcal{N}(\mu + s\Sigma\, \nabla_{x_t} \log p_\phi(y|x_t), \Sigma)$$

w/o guidance

w/ guidance

Classifier-free guidance

Classifier-Free Diffusion Guidance

Jonathan Ho
Google Research
jonathanho@google.com

Tim Salimans
Google Research
salimans@google.com

$$\nabla_x \log p(c \mid x) = \nabla_x \log p(x \mid c) - \nabla_x \log p(x)$$

- During training, drop out the class label.
- During sampling, mix the conditional and unconditional scores to obtain the classifier gradient.

Better generative processes

From a SDE perspective, the generative process of diffusion models can be a reverse-time SDE of various forward stochastic processes.

$$\mathrm{d}\mathbf{x} = \sqrt{\frac{\mathrm{d}\left[\sigma^2(t)\right]}{\mathrm{d}t}}\mathrm{d}\mathbf{w}. \qquad \mathrm{d}\mathbf{x} = -\frac{1}{2}\beta(t)\mathbf{x}\,\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\mathbf{w}.$$

VE-SDE
(SBM)

VP-SDE
(DPM)

Can we find the better forward process (and thus the generative process) tailored to a certain data modality of interest (e.g. image)?

Progressive Deblurring of Diffusion Models for Coarse-to-Fine Image Synthesis

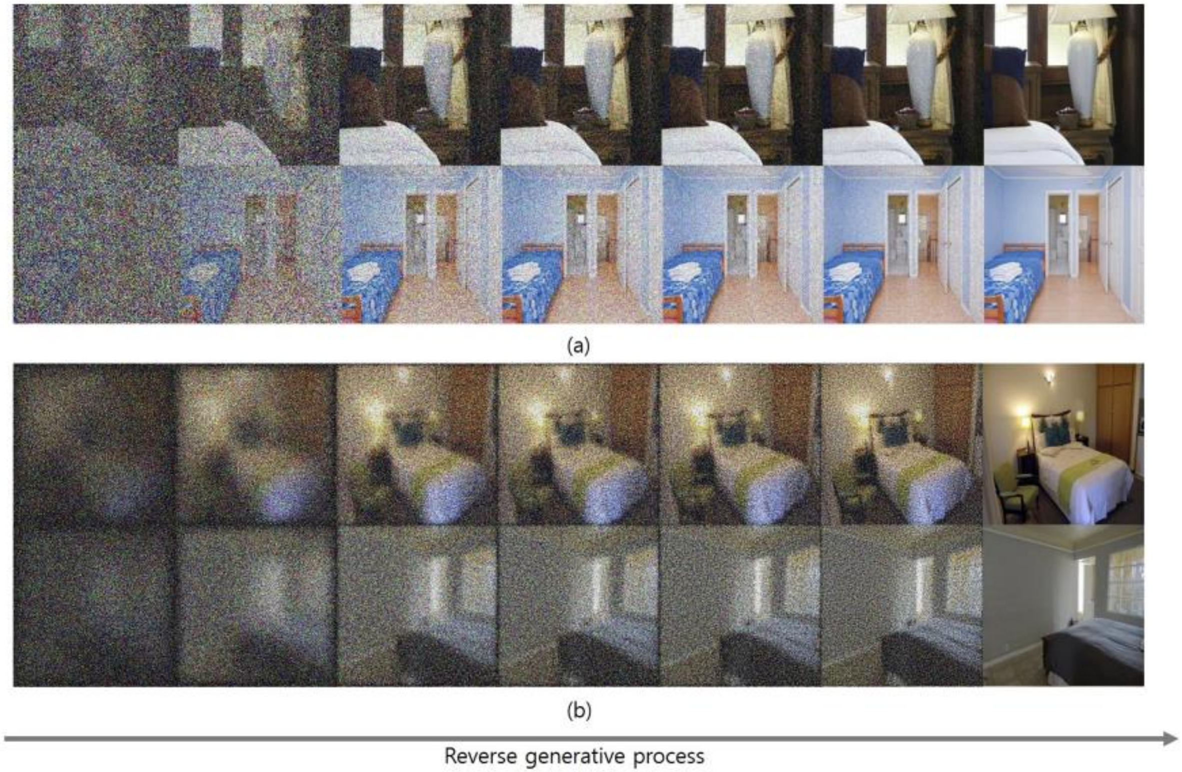
Sangyun Lee
Soongsil University
ml.swlee@gmail.com

Hyungjin Chung
Dept. of Bio and Brain Engineering
KAIST, Korea
hj.chung@kaist.ac.kr

Jaehyeon Kim
Kakao Enterprise
jay.xyz@kakaoenterprise.com

Jong Chul Ye
Kim Jaechul Graduate School of AI
KAIST, Korea
jong.ye@kaist.ac.kr

Blur diffusion



Generate images by progressive deblurring

Generalized diffusion in the rotated coordinate system

Vanilla forward diffusion $\mathbf{x}_i = \sqrt{1-\beta_i}\mathbf{x}_{i-1} + \sqrt{\beta_i}\mathbf{z}_i,$

Generalized forward diffusion $q(\bar{\mathbf{x}}_i|\bar{\mathbf{x}}_{i-1}) = \mathcal{N}(\bar{\mathbf{x}}_i; (\mathbf{I}-\mathbf{B}_i)^{\frac{1}{2}}\bar{\mathbf{x}}_{i-1}, \mathbf{B}_i\mathbf{I}),\ \ \bar{\mathbf{x}} := \mathbf{U}^T\mathbf{x}.$

Still a linear diffusion, can be trained efficiently

$$q(\bar{\mathbf{x}}_i|\bar{\mathbf{x}}_0) = \mathcal{N}(\bar{\mathbf{x}}_i; \bar{\mathbf{A}}_i^{\frac{1}{2}}\bar{\mathbf{x}}_0, (\mathbf{I}-\bar{\mathbf{A}}_i)),\ \ \mathbf{A}_i := \mathbf{I}-\mathbf{B}_i, \text{ and } \bar{\mathbf{A}}_i := \prod_{j=1}^{i}\mathbf{A}_j,$$

$$\mathbf{x}_i = \mathbf{U}\bar{\mathbf{A}}_i^{\frac{1}{2}}\mathbf{U}^T\mathbf{x}_0 + \mathbf{U}(\mathbf{I}-\bar{\mathbf{A}}_i)^{\frac{1}{2}}\mathbf{U}^T\epsilon,$$

Blur diffusion

Forward **noise** process $\mathbf{x}_i = \sqrt{1-\beta_i}\mathbf{x}_{i-1} + \sqrt{\beta_i}\mathbf{z}_i,$

$$\mathbf{x}_i = \mathbf{x}_{i-1} - (1 - \sqrt{1-\beta_i})\mathbf{x}_{i-1} + \sqrt{\beta_i}\mathbf{z}_i,$$

Forward **blur** process $q(\mathbf{x}_i|\mathbf{x}_{i-1}) = \mathcal{N}(\mathbf{x}_i; \sqrt{1-\beta_i}\mathbf{W}_i\mathbf{x}_{i-1}, \mathbf{C}_i), \mathbf{W} = \tilde{\mathbf{U}}\mathbf{D}\tilde{\mathbf{U}}^T$

$$\mathbf{x}_i = \mathbf{x}_{i-1} - \mathbf{H}(\mathbf{x}_{i-1}, i-1) + \mathbf{C}_i^{\frac{1}{2}}\mathbf{z}_i,$$

Blur matrix

where $\mathbf{H}(\mathbf{x}_i, i) = \mathbf{x}_i - \sqrt{1-\beta_{i+1}}\mathbf{W}_{i+1}\mathbf{x}_i$ is an unnormalized Gaussian high-pass filter.

Blur diffusion is a special case of the generalized diffusion when $\mathbf{B}_i = \mathbf{I} - (1-\beta_i)\mathbf{D}^{2f(i)}$ *and* $\mathbf{U} = \tilde{\mathbf{U}}$.

Blur diffusion

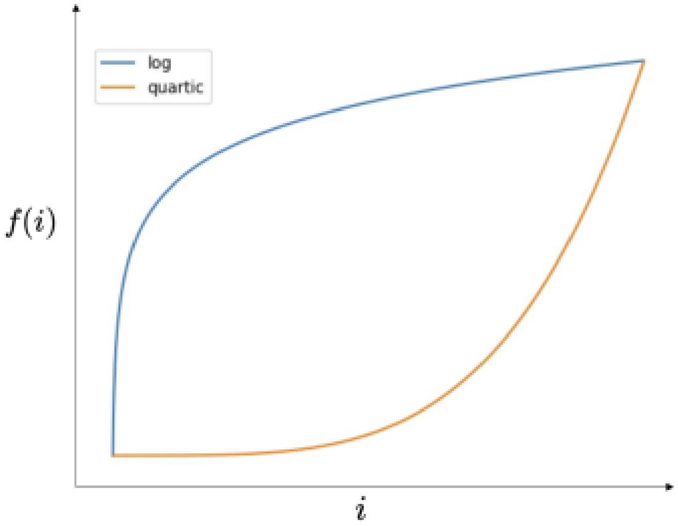
$$\mathbf{x}_{i-1} = \mathbf{x}_i - (\sqrt{1-\beta_{i+1}} - 1)\mathbf{x}_i + \beta_{i+1}\mathbf{s}_\theta(\mathbf{x}_i, i) + \sqrt{\beta_{i+1}}\mathbf{z}_i.$$

Reverse generative process

$$\mathbf{x}_{i-1} = \mathbf{x}_i + \mathbf{H}(\mathbf{x}_i, i)\mathbf{x}_i - \mathbf{U}\mathbf{B}_{i+1}(\mathbf{I} - \bar{\mathbf{A}}_i)\mathbf{U}^T\boldsymbol{\epsilon}_\theta(\mathbf{x}_i, i) + \mathbf{U}\mathbf{B}_{i+1}^{\frac{1}{2}}\mathbf{U}^T\mathbf{z}_i,$$

Blur diffusion

$f(N)$	f_type	FID-10K bedroom	FID-10K church
0 (w/o blur)	N/A	9.24	6.04
0.6	log	73.23	
0.14	quartic	**7.86**	**5.89**



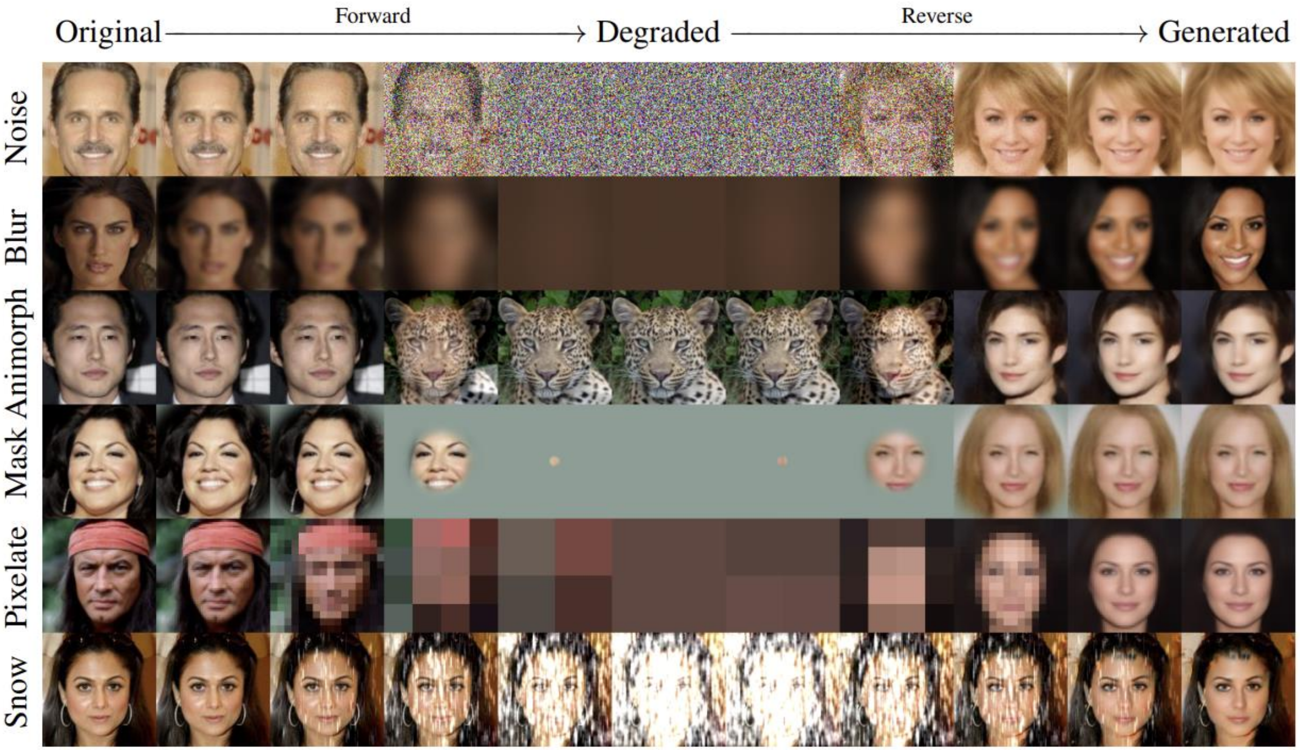
Blur diffusion

$\mathbf{D} := \mathbf{I} - \mathbf{D}$: fine-to-coarse generation

Figure 3: Comparison of generated images with different generation strategies. Left: fine-to-coarse, right: coarse-to-fine.

Inductive bias matters!

Other forward processes



Cold Diffusion: Inverting Arbitrary Image Transforms Without Noise

Conclusion

- Diffusion models can be viewed as the discretization of SDEs.
- DPMs and SBMs (Song & Ermon) are the discretizations of VP and VE SDEs.
- Other forward processes can also be considered (e.g. blur).

References

- Probabilistic Machine Learning: Advanced Topics
- Deep Learning Lecture Series 2020 (https://www.deepmind.com/learning-resources/deep-learning-lecture-series-2020)
- Generative Modeling by Estimating Gradients of the Data Distribution - Stefano Ermon (https://youtu.be/8TcNXi3A5DI)

Thanks!