Diffusion Models

Lunit Research Seminar 22.03.11
Geonwoon Jang

In this presentation...

Following the line of **Diffusion Probabilistic Model** (=**Diffusion Model**),
Introducing essence of it and its results

- Deep Unsupervised Learning using Non-Equilibrium Thermodynamics (ICML ‘15)
  - Beginning
  - Mostly for theoretical background, derivation, related materials
- Denoising Diffusion Probabilistic Models (NeurIPS ‘20)
  - Improvement, simplified, more things to look into
  - Overall notations and expressions are neater
- Diffusion Models Beat GANs on Image Synthesis (NeurIPS ‘21)
  - Mostly for good results, not much for theory and method in here

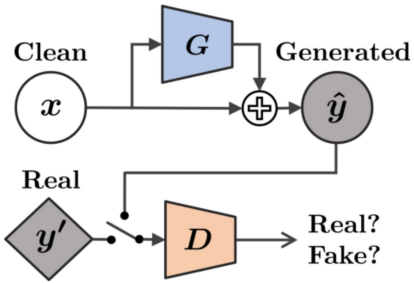
In this presentation...

Following the line of **Diffusion Probabilistic Model** (=**Diffusion Model**),
Introducing essence of it and its results

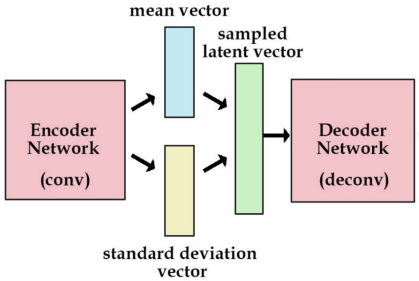
- Deep Unsupervised Learning using Non-Equilibrium Thermodynamics (ICML ‘15)
  DPM
- Denoising Diffusion Probabilistic Models (NeurIPS ‘20)
  DDPM
- Diffusion Models Beat GANs on Image Synthesis (NeurIPS ‘21)
  ADM

Motivation

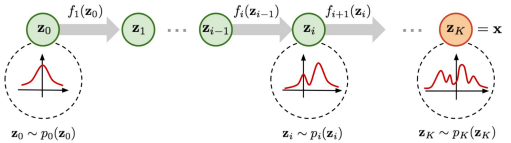
Thinking of (neural) Generative Models,



GAN-based



VAE-based



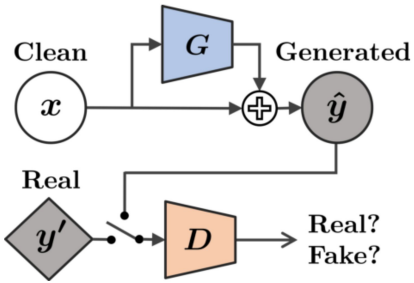
Flow-based

How to evaluate/supervise the generative distribution accurately?

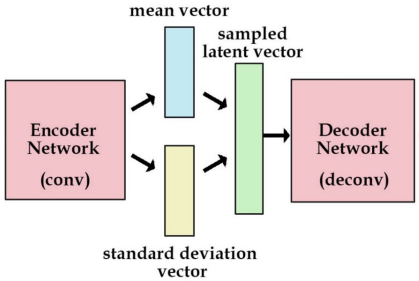
How to reason and see what's happening in there?

Motivation

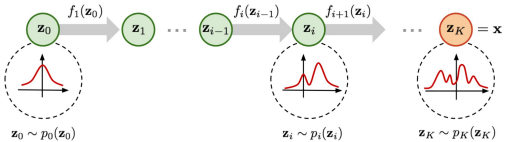
Thinking of (neural) Generative Models,



GAN-based



VAE-based



Flow-based

How to evaluate/supervise the generative distribution accurately?

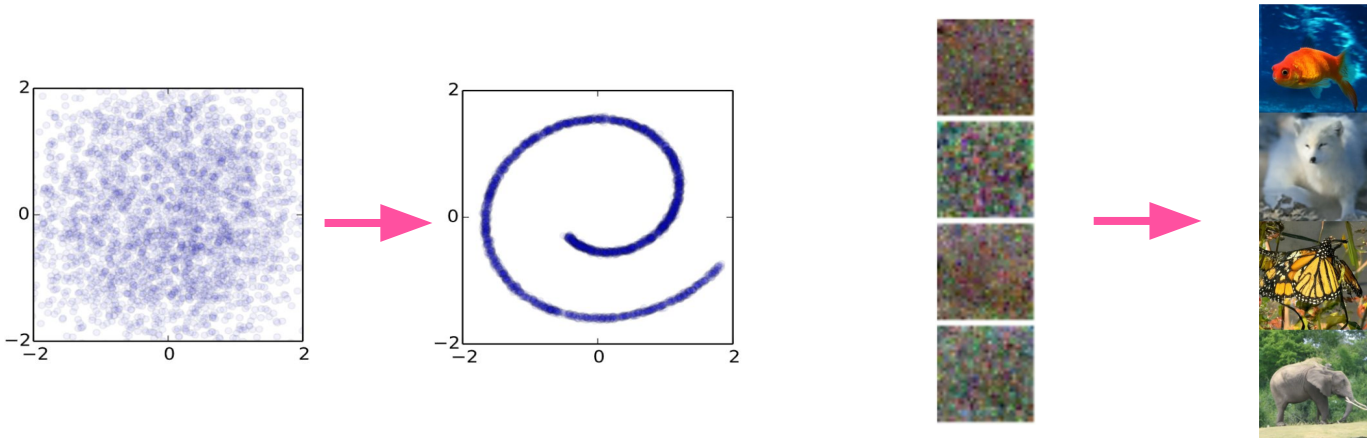
How to reason and see what's happening in there?

Some sharing points:

Density estimation by tractable & invertible transforms (micro-steps)

Motivation

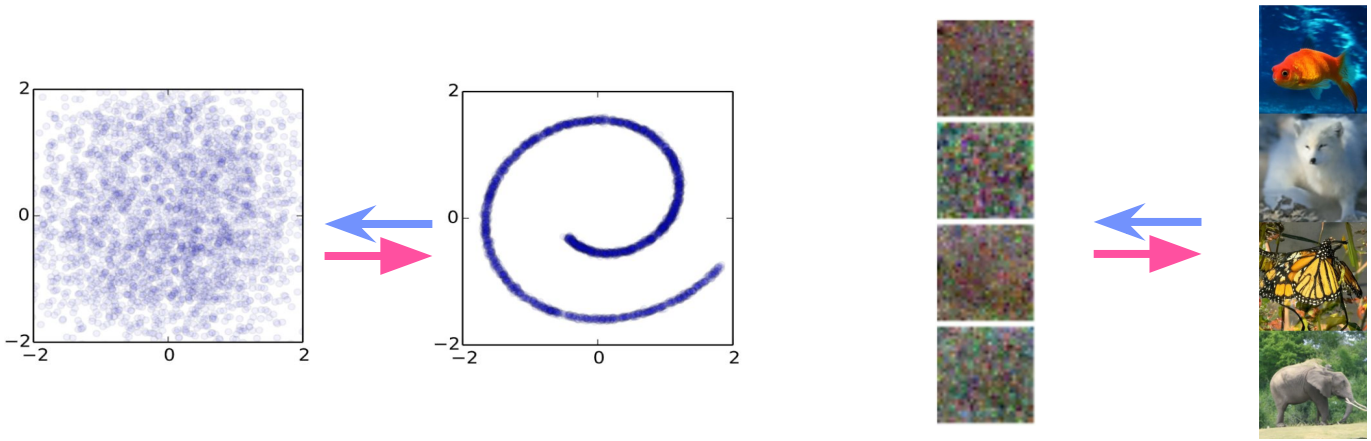
What should generative models do?



Transition from a **well-known distribution** (ex: Gaussian) to desired **data distribution**

Motivation

What should generative models do?

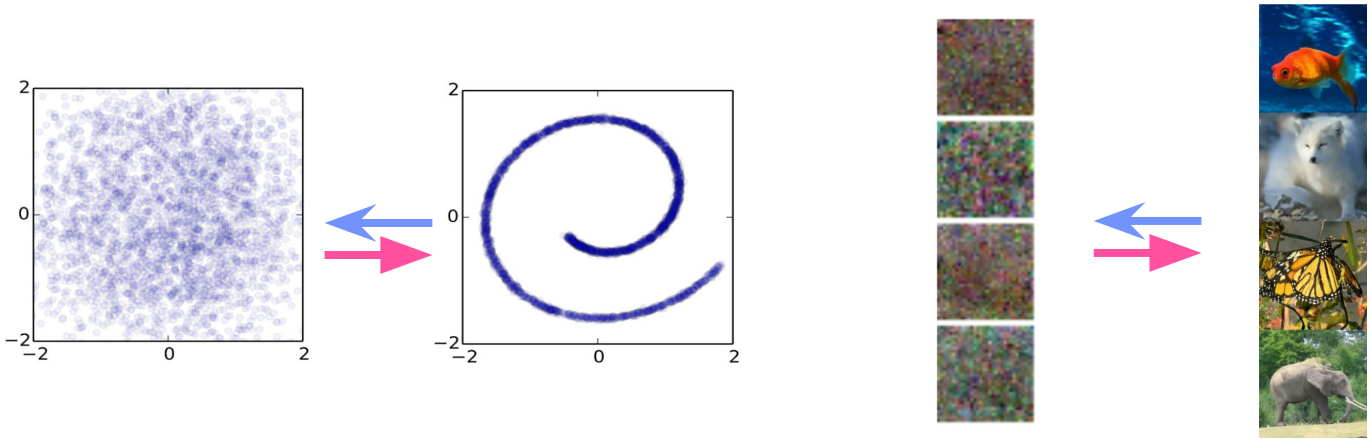


Transition from a **well-known distribution**(ex: Gaussian) to desired **data distribution**

Reverse process of making **data distribution** into **Gaussian distribution**

Motivation

Thermodynamically speaking,



Transition from a **well-known distribution**(ex: Gaussian) to desired **data distribution**

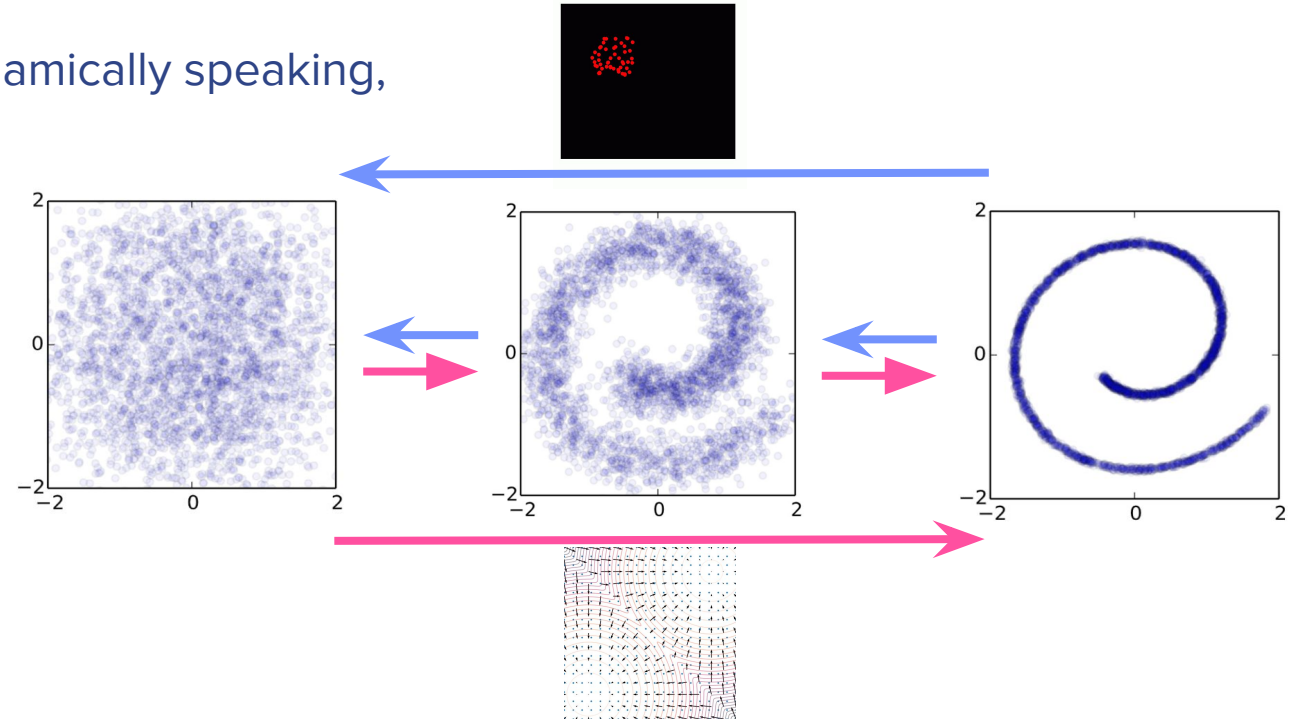
Reverse process of making **data distribution** into **Gaussian distribution**

Reverse process of transitioning from **non-equilibrium** to **equilibrated** states

Low *H* **High *H***

Motivation

Thermodynamically speaking,

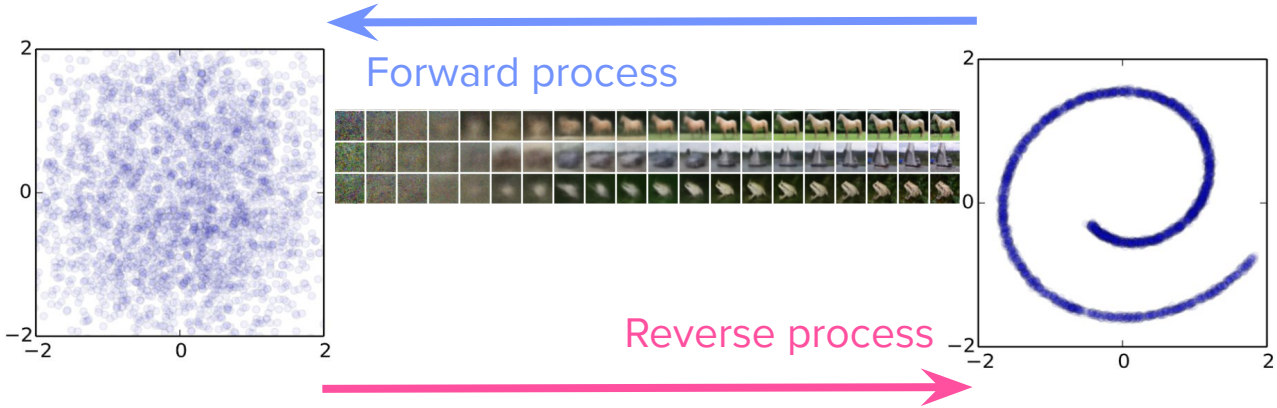


Reverse process of transitioning from **non-equilibrium** to **equilibrated** states

Low *H* **High *H***

Idea from StatPhys

But how to “reverse” it?

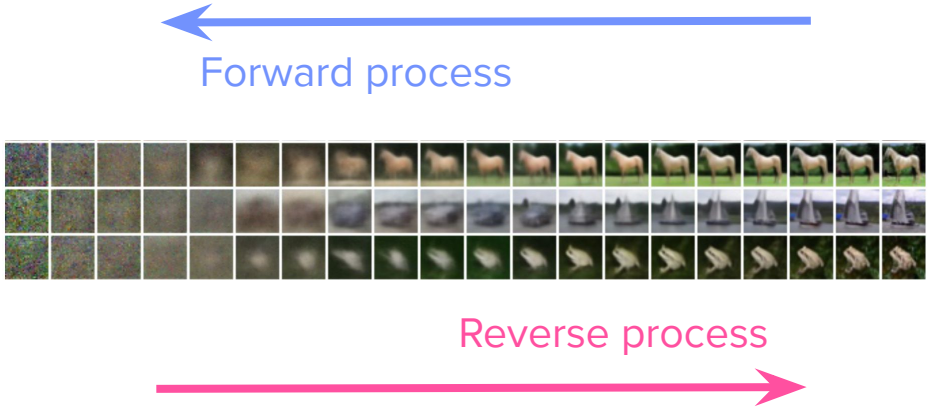


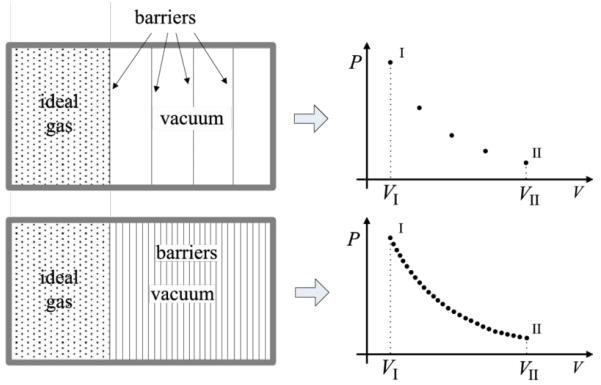
1. Set the forward process as **diffusion process**

   Adding noise gradually, micro-step repeatedly

Idea from StatPhys

But how to “reverse” it?





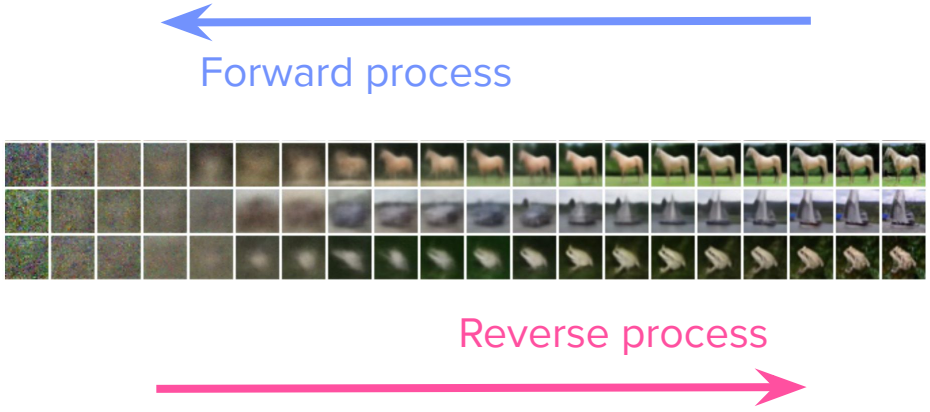
1. Set the forward process as diffusion process
2. If the process is **quasi-static**, its reverse process can be **expressed in the same form**.

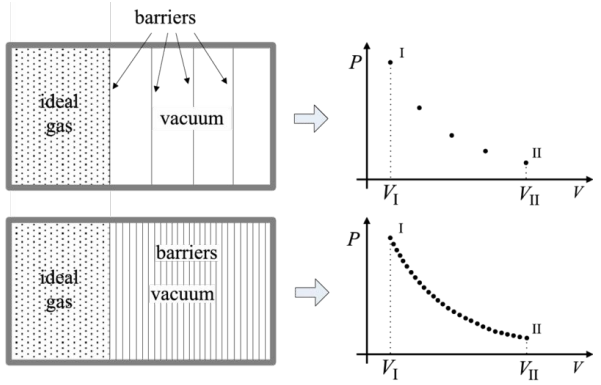
Micro-changes taken infinitely slowly

If we take **forward(diffusion) kernels as Gaussian**, their **reverse kernels can also be Gaussian**

Idea from StatPhys

But how to “reverse” it?

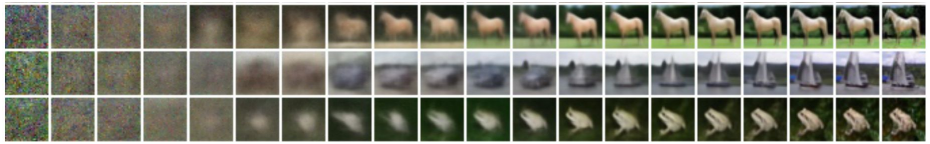




1. Set the forward process as diffusion process with Gaussian
2. If the process is quasi-static, its reverse process can also be Gaussian
3. **Predict μ and Σ** for each steps in reverse process

Method

What'd be the result, and why is it beneficial?

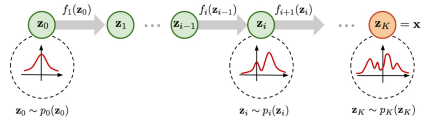


Reverse process: **Generator**

1. Forward process as diffusion process with Gaussian
2. Reverse process also being Gaussian
3. Predict μ and Σ for each steps in reverse process

"Since each step in the diffusion chain has an analytically evaluable probability, the full chain can also be analytically evaluated."

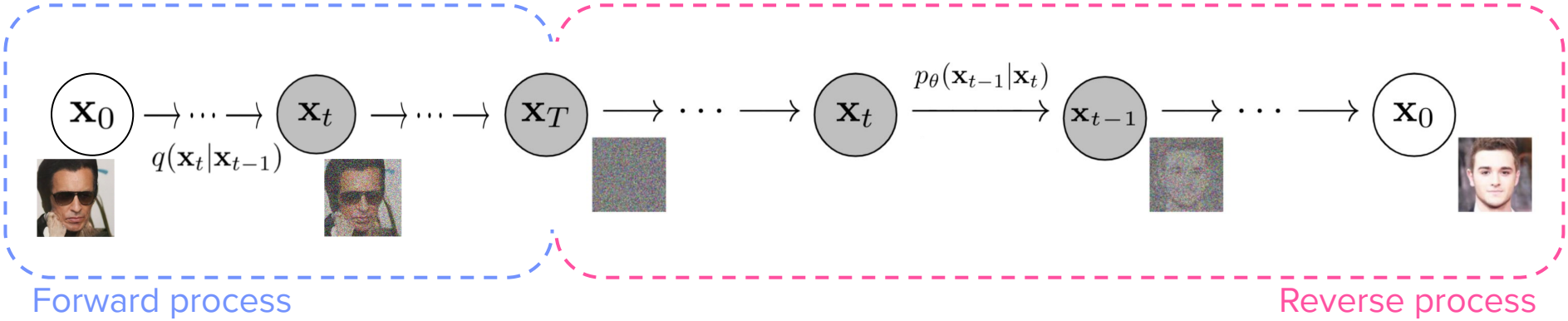
"Estimating small perturbations is more tractable than explicitly describing the full distribution with a single, non-analytically-normalizable function."



(Also good to check: Normalizing Flow)

- Behaves as Generator
- The kernels are all Gaussian -> Tractable -> Can evaluate/supervise in generative distribution
- Can check/evaluate/control the intermediate steps

Method - Overall and Notation

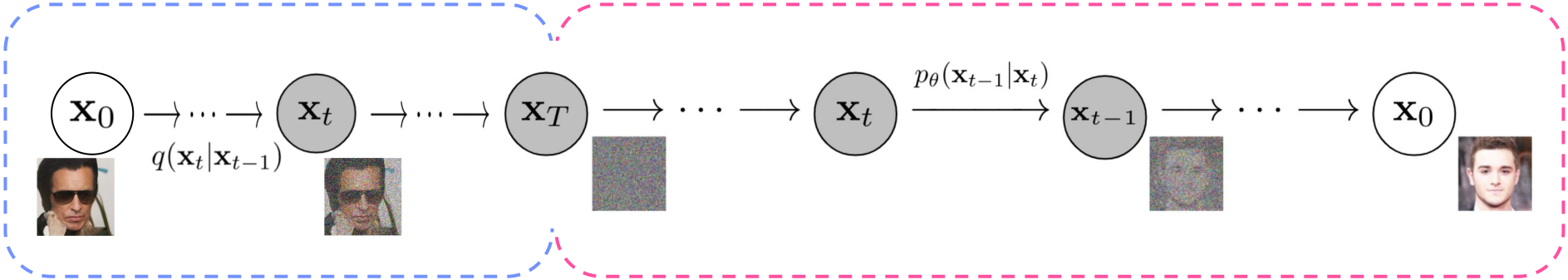


x_0: At timestamp 0, Data sample.

x_T: At timestamp T, Gaussian noise sample (that underwent all forward process).

x_t: At timestamp t.

Method - Overall and Notation



Forward process

Reverse process

q: In forward process(diffusion process, forward trajectory)

$q(x_t)$: Distribution at forward timestamp t.

$X_0 \sim q(x_0)$

$q(x_t|x_{t-1})$: Transition of each forward process step.

$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t \mathrm{I})$

β_t: Diffusion step size. Small enough, can be learned (or not, also okay)

$q(x_{0:T}) = q(x_0)\prod_{t=1}^{T} q(x_t|x_{t-1})$: Joint distribution of whole forward process.

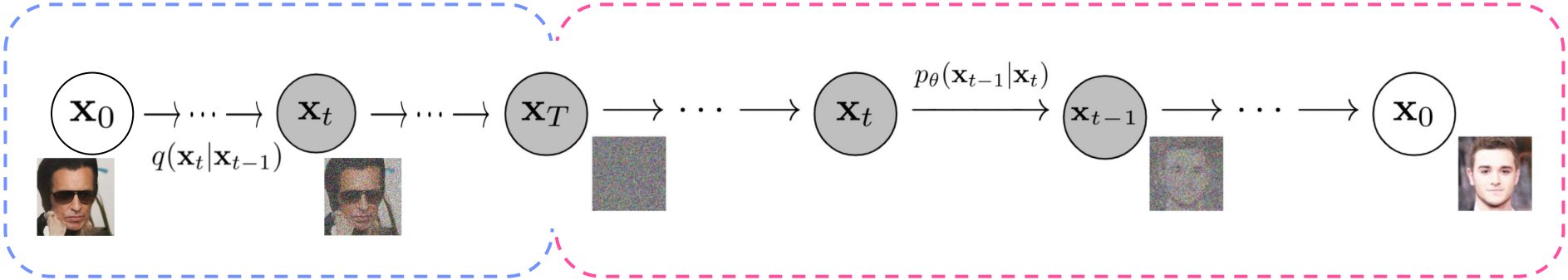
(At arbitrary time t,

$$q(x_t|x_0) = \mathcal{N}\left(x_t; \sqrt{\overline{\alpha_t}}x_0, (1-\overline{\alpha_t})\mathrm{I}\right),$$

$$\alpha_t = 1-\beta_t, \overline{\alpha_t} = \prod^{t} \alpha_s$$

will be visited later)

Method - Overall and Notation



p_θ, p: In reverse process(reverse diffusion process, backward trajectory). Generative distribution.

$p(x_t)$: Distribution at reverse timestamp t.

$$X_T \sim p(x_T) = \mathcal{N}(x_T; 0, \mathrm{I})$$

$p(x_{t-1}|x_t)$: Transition of each reverse process step.

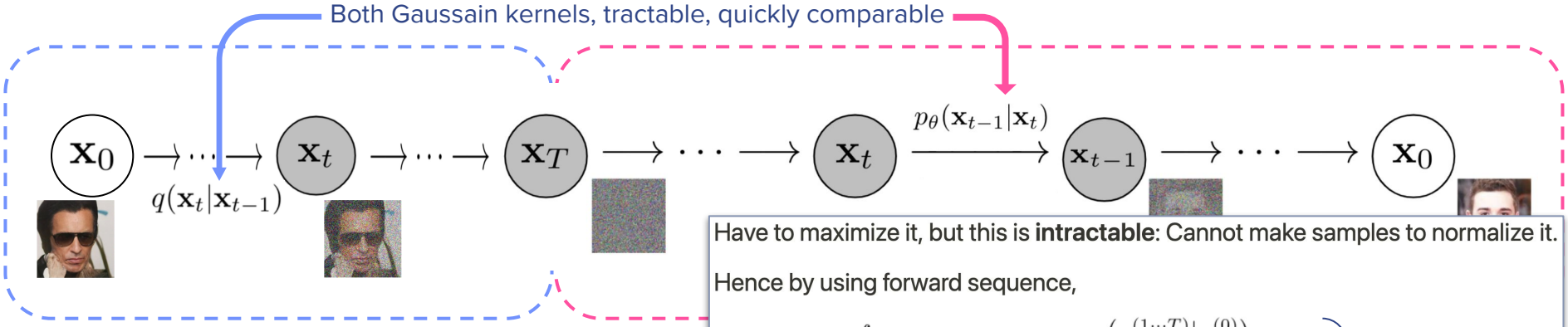
$$p(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

$\mu_\theta(x_t, t)$ and $\Sigma_\theta(x_t, t)$: Implemented with CNN

Being slow enough,
forward kernels as Gaussian
-> reverse kernels can be Gaussian

$p(x_{T:0}) = p(x_T) \prod_{t=1}^{T} p(x_{t-1}|x_t)$: Joint distribution of whole reverse process.

Method - Learning Objective



We'd like to **maximize the likelihood** of the backward proc

$$\underbrace{\log p_\theta(x_0)}_{:= \text{Evidence}} = \log \left[\int p(x_{T:0}) dx_{T:1} \right]$$

Have to maximize it, but this is **intractable**: Cannot make samples to normalize it.

Hence by using forward sequence,

$$p\left(\mathbf{x}^{(0)}\right) = \int d\mathbf{x}^{(1\cdots T)} p\left(\mathbf{x}^{(0\cdots T)}\right) \frac{q\left(\mathbf{x}^{(1\cdots T)}|\mathbf{x}^{(0)}\right)}{q\left(\mathbf{x}^{(1\cdots T)}|\mathbf{x}^{(0)}\right)}$$

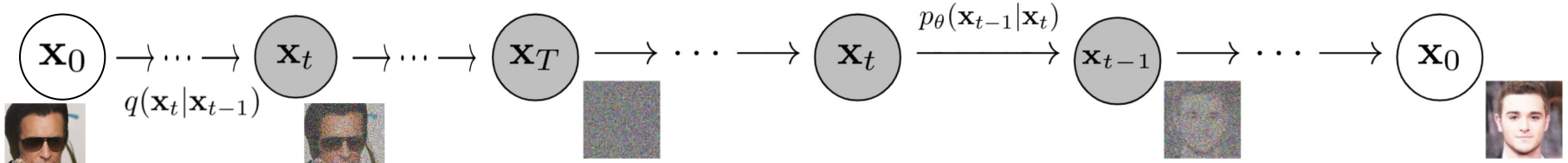
$$= \int d\mathbf{x}^{(1\cdots T)} q\left(\mathbf{x}^{(1\cdots T)}|\mathbf{x}^{(0)}\right) \frac{p\left(\mathbf{x}^{(0\cdots T)}\right)}{q\left(\mathbf{x}^{(1\cdots T)}|\mathbf{x}^{(0)}\right)}$$

$$= \int d\mathbf{x}^{(1\cdots T)} q\left(\mathbf{x}^{(1\cdots T)}|\mathbf{x}^{(0)}\right) \cdot p\left(\mathbf{x}^{(T)}\right) \prod_{t=1}^{T} \frac{p\left(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}\right)}{q\left(\mathbf{x}^{(t)}|\mathbf{x}^{(t-1)}\right)}.$$

Annealed Importance Sampling

(You may have seen similar derivation for Variational Inference in VAE)

Method - Learning Objective



We'd like to **maximize the likelihood** of the backward process result, x_0.

$$\underbrace{\log p_\theta(x_0)}_{:= \text{Evidence}} = \log \left[\int p(x_{T:0}) dx_{T:1} \right]$$

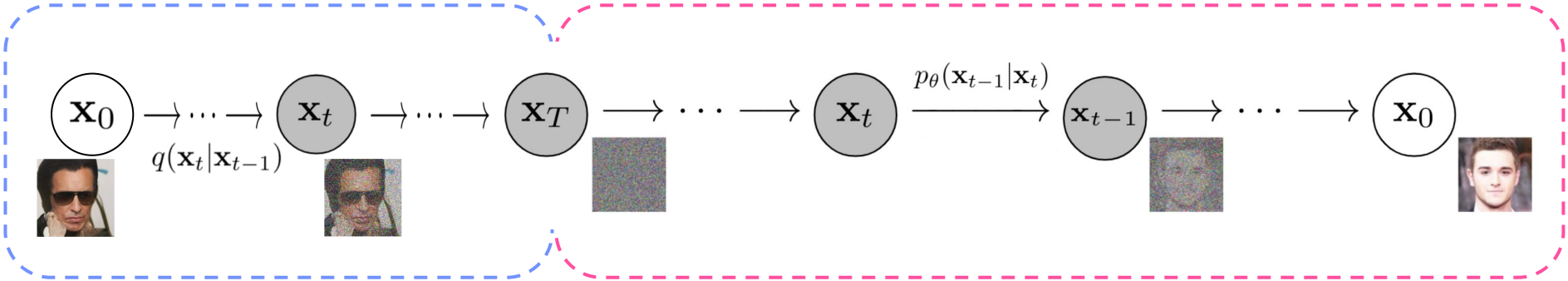
Now getting its ELBO(Evidence Lower Bound),

$$L \geq \int d\mathbf{x}^{(0\cdots T)} q\left(\mathbf{x}^{(0\cdots T)}\right) \cdot \log \left[p\left(\mathbf{x}^{(T)}\right) \prod_{t=1}^{T} \frac{p\left(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}\right)}{q\left(\mathbf{x}^{(t)}|\mathbf{x}^{(t-1)}\right)} \right].$$

$$\int d\mathbf{x}^{(1\cdots T)} q\left(\mathbf{x}^{(1\cdots T)}|\mathbf{x}^{(0)}\right) \cdot p\left(\mathbf{x}^{(T)}\right) \prod_{t=1}^{T} \frac{p\left(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}\right)}{q\left(\mathbf{x}^{(t)}|\mathbf{x}^{(t-1)}\right)}.$$

(You may have seen similar derivation for Variational Inference in VAE)

Method - Learning Objective



Getting rid of former term,

$$\log p(x_T) + \sum_{t \geq 1} \log \frac{p(x_{t-1}|x_t)}{q(x_t|x_{t-1})}$$

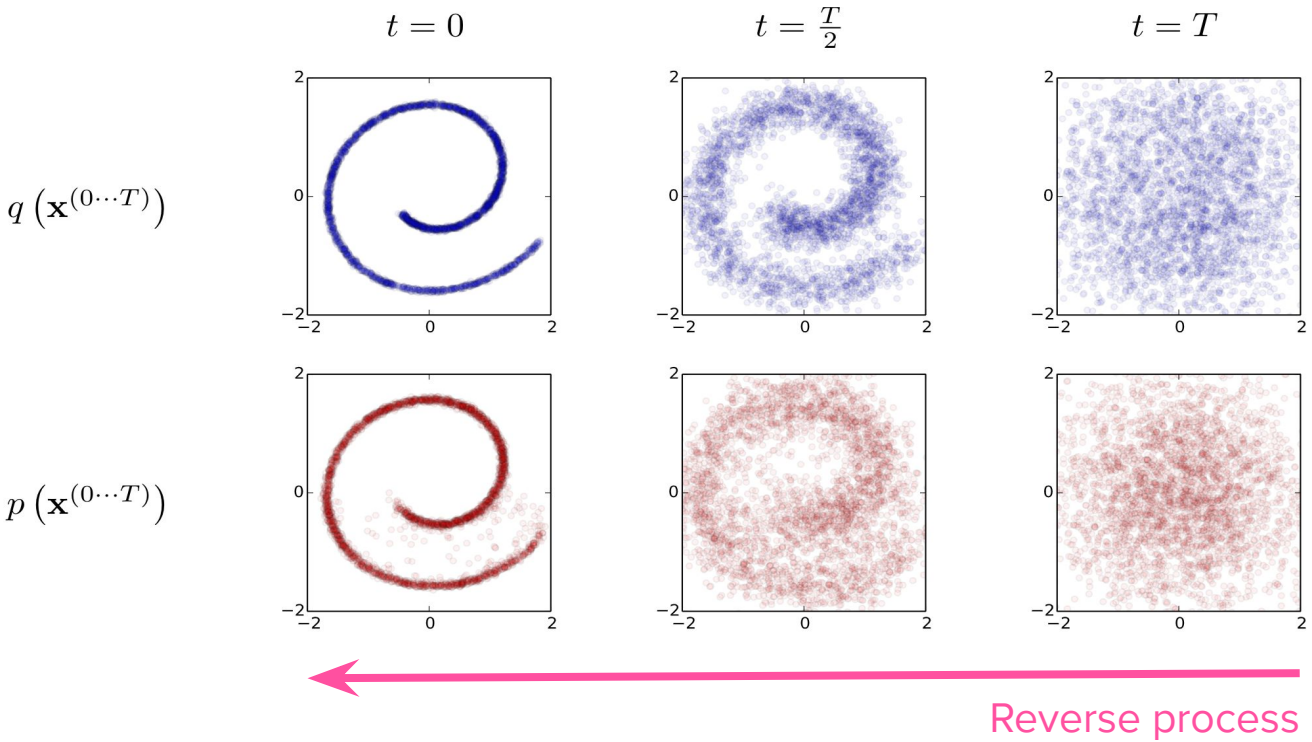
Taking negative, **Minimize**:

$$\mathbb{E}\left[-\log p_\theta(x_0)\right] \leq \mathbb{E}\left[-\log p(x_T) - \sum_{t \geq 1} \log \frac{p(x_{t-1}|x_t)}{q(x_t|x_{t-1})}\right]$$

(You may have seen similar derivation for Variational Inference in VAE)

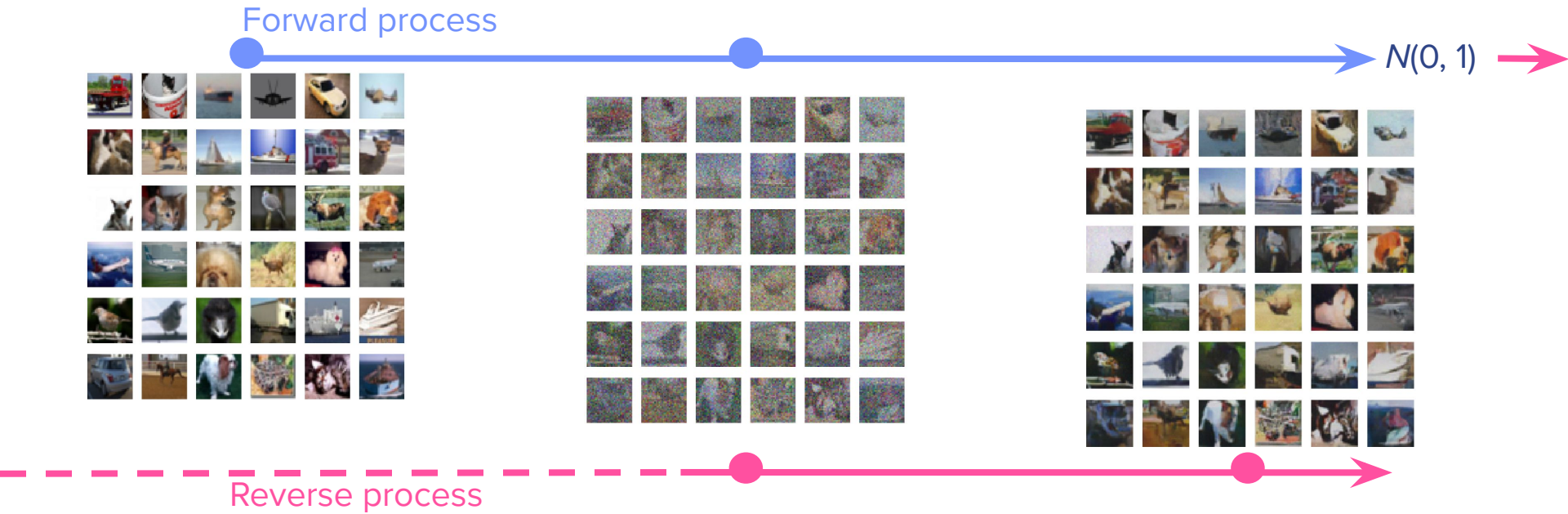
Taking a break: Some Results of DPM(2015)

Toy exp: Swiss-roll



Taking a break: Some Results of DPM(2015)

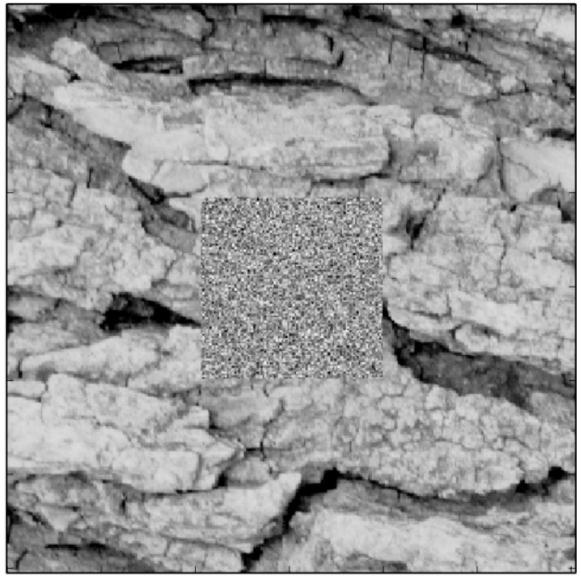
CIFAR-10: Denoising again



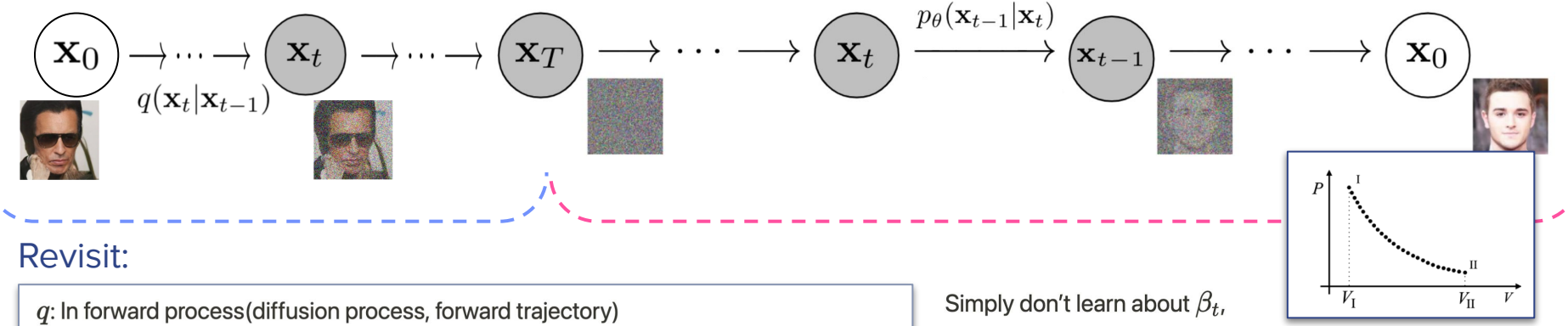
Taking a break: Some Results of DPM(2015)

Generation

$N(0, 1)$ Reverse process



DDPM(2020) Improvements



Revisit:

q: In forward process(diffusion process, forward trajectory)

$q(x_t)$: Distribution at forward timestamp t.

$X_0 \sim q(x_0)$

$q(x_t|x_{t-1})$: Transition of each forward process step.

$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t \mathrm{I})$

β_t: Diffusion step size. Small enough, can be learned (or not, also okay)

$q(x_{0:T}) = q(x_0)\prod_{t=1}^{T} q(x_t|x_{t-1})$: Joint distribution of whole forward process.

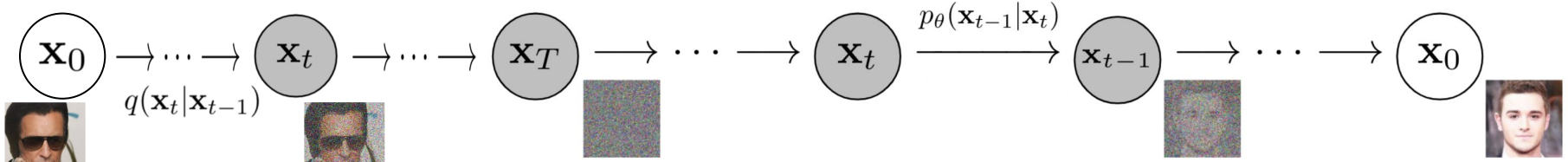
Simply don't learn about β_t,

just set them as $\beta_1 = 10^{-4}, \beta_T = 0.02$ linearly.

Actually, more intuitively convincing!

: If it's indeed like a
quasi-static diffusion process,
it should be okay w.o. carefully tuning them.

DDPM(2020) Improvements



Revisit:

q: In forward process(diffusion process, forward trajectory)

$q(x_t)$: Distribution at forward timestamp t.

$X_0 \sim q(x_0)$

$q(x_t|x_{t-1})$: Transition of each forward process step.

$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t \mathrm{I})$

β_t: Diffusion step size. Small enough, can be learned (or not, also okay)

$q(x_{0:T}) = q(x_0)\prod_{t=1}^{T} q(x_t|x_{t-1})$: Joint distribution of whole forward process.

Now β_t are constants and

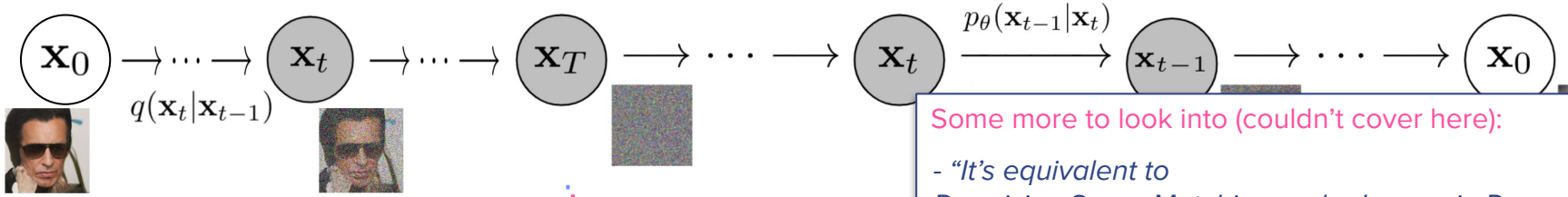
At arbitrary time t,

$$q(x_t|x_0) = \mathcal{N}\left(x_t; \sqrt{\overline{\alpha_t}}x_0, (1-\overline{\alpha_t})\mathrm{I}\right),$$
$$\alpha_t = 1-\beta_t, \overline{\alpha_t} = \prod^{t} \alpha_s$$

So, all x_t can be expressed with x_0 and t, as:

$$x_t(x_0, \epsilon) = \sqrt{\overline{\alpha_t}}x_0 + \sqrt{1-\overline{\alpha_t}}\epsilon,$$
$$\epsilon \sim \mathcal{N}(0, \mathrm{I})$$

DDPM(2020) Improvements



Some more to look into (couldn't cover here):

- *"It's equivalent to Denoising Score-Matching under Langevin Dynamics"*** -> Finite-time sampling chain under Langevin dynamics.

$$\dot{x} = f(x) + \eta$$

Related work: DDIM(2021)* -> Provides sampling method that reduces T.

Revisit:

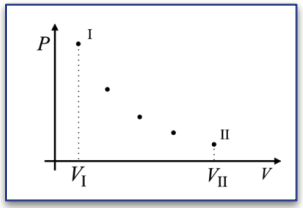
$$\mathbb{E}\left[-\log p_\theta(x_0)\right] \leq \mathbb{E}\left[-\log p(x_T) - \sum_{t\geq 1} \log \frac{p(x_{t-1}|x_t)}{q(x_t|x_{t-1})}\right]$$

Here only variables are $\mu_\theta(x_0, t)$.

So we can reduce the objective, with some reparameterization and cleanup.....(omitted):

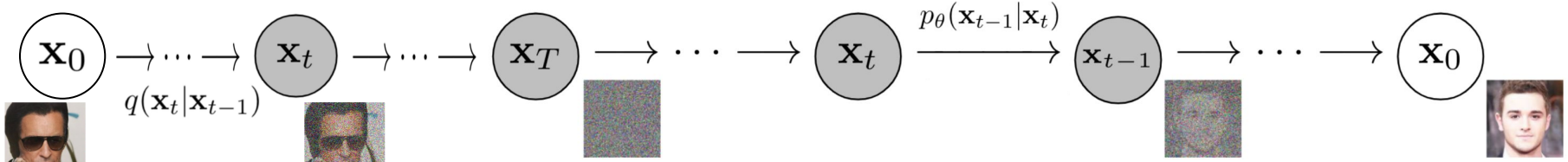
$$\mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2\right]$$

($\epsilon_\theta(\cdot)$: CNN we train, still a function of (x_0, t). $\epsilon \sim \mathcal{N}(0, \mathrm{I})$)



* Song, Jiaming, et. al., "Denoising Diffusion Implicit Models", 2021
** Song, Yang, et. al., "Improved Techniques for Training Score-Based Generative Models", 2021

DDPM(2020) Improvements



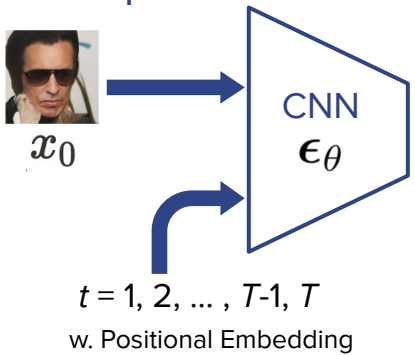
Final simplified objective:

$$\mathbb{E}_{x_0,\epsilon}\left[\left\|\epsilon-\epsilon_\theta(\sqrt{\overline{\alpha_t}}x_0,\sqrt{1-\overline{\alpha_t}}\epsilon,t)\right\|^2\right]$$

$$\mu_\theta(x_0,t)=\frac{1}{\sqrt{\alpha_t}}\left(\frac{\beta_t}{\sqrt{1-\overline{\alpha_t}}}\epsilon_\theta(x_t,t)\right)$$

DDPM(2020) Improvements

Now it's like this simplified:

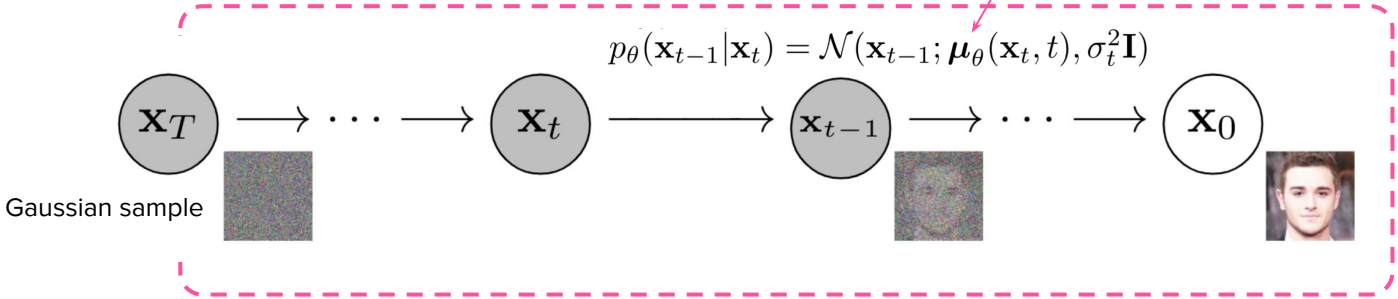


Loss:

$$\mathbb{E}_{x_0,\epsilon}\left[\left\|\epsilon - \epsilon_\theta(\sqrt{\overline{\alpha_t}}x_0, \sqrt{1-\overline{\alpha_t}}\epsilon, t)\right\|^2\right]$$

And to generate a sample:

$$\mu_\theta(x_0,t) = \frac{1}{\sqrt{\alpha_t}}\left(\frac{\beta_t}{\sqrt{1-\overline{\alpha_t}}}\epsilon_\theta(x_t,t)\right)$$



Results of DDPM(2020)

Table 1: CIFAR10 results. NLL measured in bits/dim.

Model	IS	FID	NLL Test (Train)
Conditional			
EBM [11]	8.30	37.9	
JEM [17]	8.76	38.4	
BigGAN [3]	9.22	14.73	
StyleGAN2 + ADA (v1) [29]	**10.06**	**2.67**	
Unconditional			
Diffusion (original) [53]			≤ 5.40
Gated PixelCNN [59]	4.60	65.93	3.03 (2.90)
Sparse Transformer [7]			**2.80**
PixelIQN [43]	5.29	49.46	
EBM [11]	6.78	38.2	
NCSNv2 [56]		31.75	
NCSN [53]	8.87 ± 0.12	25.32	
SNGAN [39]	8.22 ± 0.05	21.7	
SNGAN-DDLS [4]	9.09 ± 0.10	15.42	
StyleGAN2 + ADA (v1) [29]	$\mathbf{9.74} \pm 0.05$	3.26	
Ours (L, fixed isotropic $\mathbf{\Sigma}$)	7.67 ± 0.13	13.51	≤ 3.70 (3.69)
Ours (L_{simple})	9.46 ± 0.11	**3.17**	≤ 3.75 (3.72)

Results of DDPM(2020)

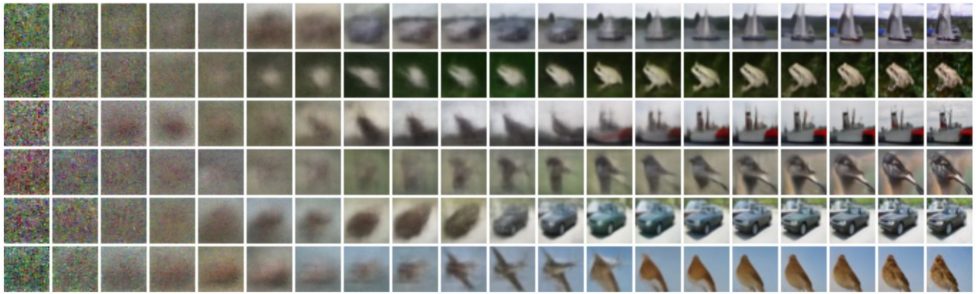
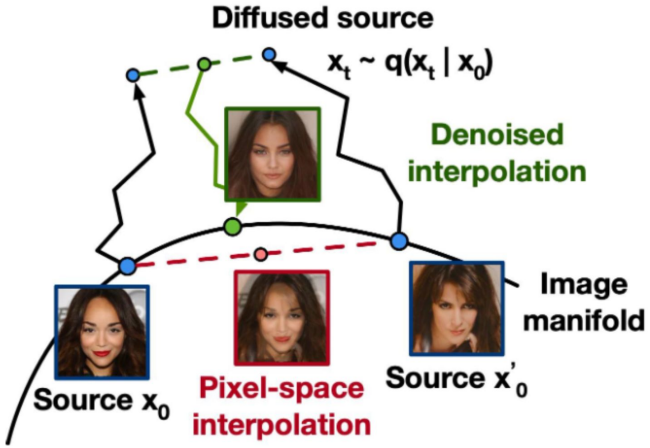


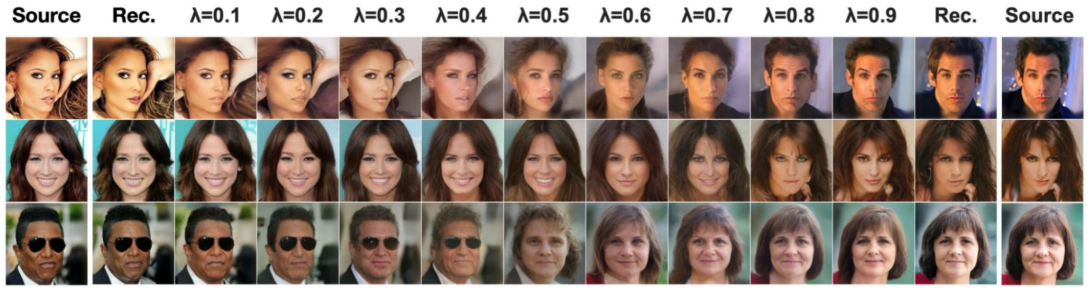
Figure 14: Unconditional CIFAR10 progressive generation

Results of DDPM(2020)

Interpolating in encoded latent

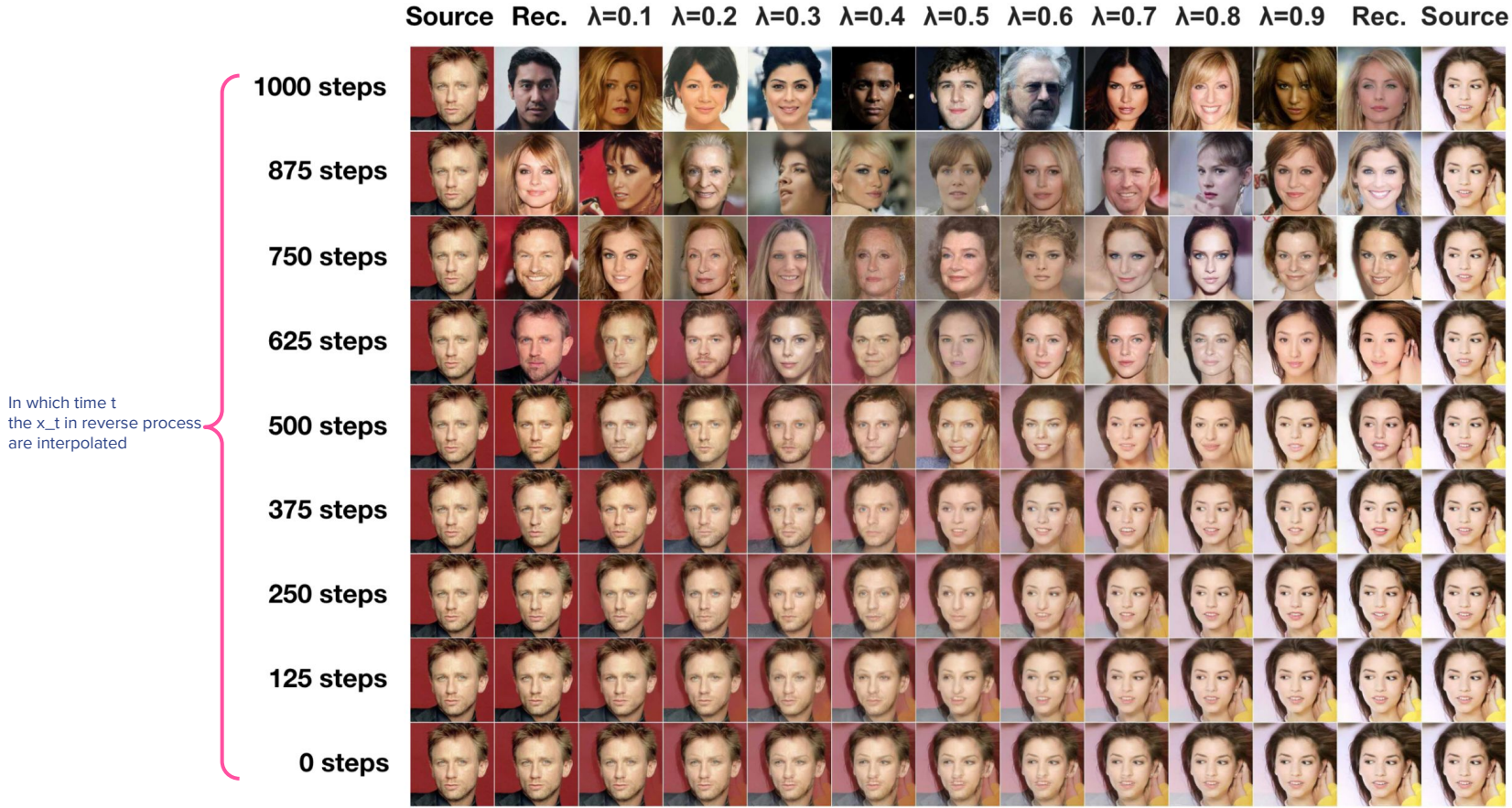
- Whether forward diffusion process behave as desired
- Whether the generator successfully reverse it





Interpolated at t=500 in reverse process

Results of DDPM(2020)



ADM(2021) Improvement

(Rather keep it simple, omit details and ablation studies, to just introduce the results)

Guiding each timestamps' intermediate *x_t* class-conditioned:

$$\mu_\theta(x_t) \rightarrow \mu_\theta(x_t) + \Sigma_\theta(x_t) \cdot s\nabla_{x_t} \log \underbrace{f(x_t)}_{\text{Trained classifier}}$$

s=1

s=10

...And application of DDIM(2021)* for fewer reverse steps(if used), and interpolation in latent

+) Other tweaks/tuning in sampling steps, Bigger architecture, More tuning, etc.

* Song, Jiaming, et. al., "Denoising Diffusion Implicit Models", 2021

Results of ADM(2021)

Comparing with SoTA: Quantitative

Model	FID	sFID	Prec	Rec
LSUN Bedrooms 256×256				
DCTransformer† [48]	6.40	6.66	0.44	**0.56**
DDPM [31]	4.89	9.07	0.60	0.45
IDDPM [49]	4.24	8.21	0.62	0.46
StyleGAN [33]	2.35	6.62	0.59	0.48
ADM (dropout)	**1.90**	**5.59**	**0.66**	0.51
LSUN Horses 256×256				
StyleGAN2 [34]	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 [31]	17.1	12.4	0.53	0.48
StyleGAN2 [34]	7.25	**6.33**	0.58	0.43
ADM (dropout)	**5.57**	6.69	**0.63**	**0.52**

Model	FID	sFID	Prec	Rec
ImageNet 64×64				
BigGAN-deep* [8]	4.06	3.96	**0.79**	0.48
IDDPM [49]	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**
ImageNet 128×128				
BigGAN-deep [8]	6.02	7.18	**0.86**	0.35
LOGAN† [74]	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

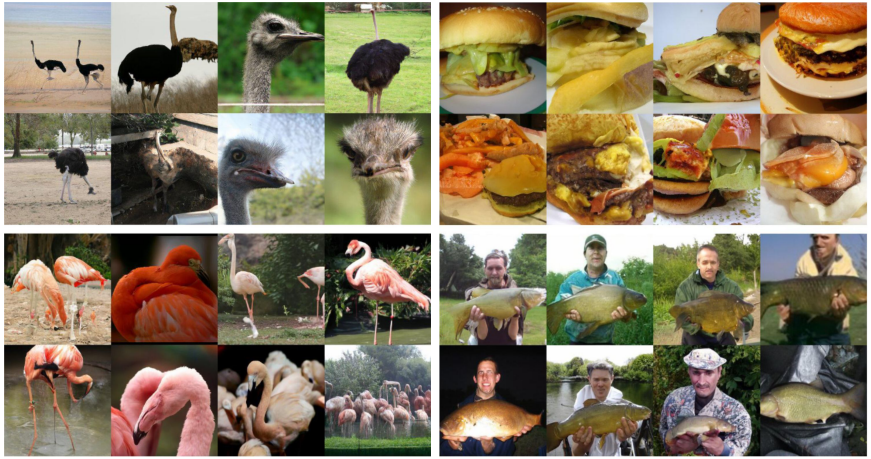
Model	FID	sFID	Prec	Rec
ImageNet 256×256				
DCTransformer† [48]	36.51	8.24	0.36	**0.67**
VQ-VAE-2†‡ [57]	31.11	17.38	0.36	0.57
VQ-VAE-2 (RS)†‡§ [57]	∼ 10			
VQ-GAN‡ [21]	15.97	19.05	0.63	0.58
VQ-GAN (RS)‡§ [21]	5.06	7.34	0.79	0.48
IDDPM‡ [49]	12.26	5.42	0.70	0.62
SR3†‡ [60]	11.30			
BigGAN-deep [8]	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 [8]	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

T=1000 for LSUN
T=250 for ImageNet, =25 if DDIM* adopted
-G: Class-guided

Results of ADM(2021)

Comparing with SoTA: Qualitative - ImageNet

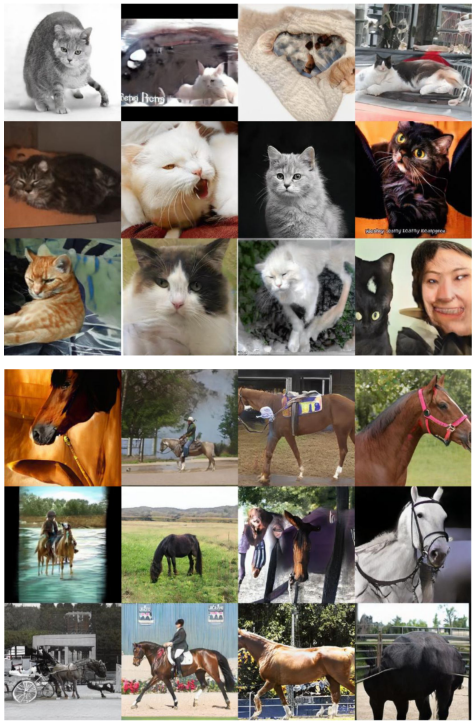
BigGAN-deep



ADM-G

Results of ADM(2021)

Comparing with SoTA: Qualitative - LSUN

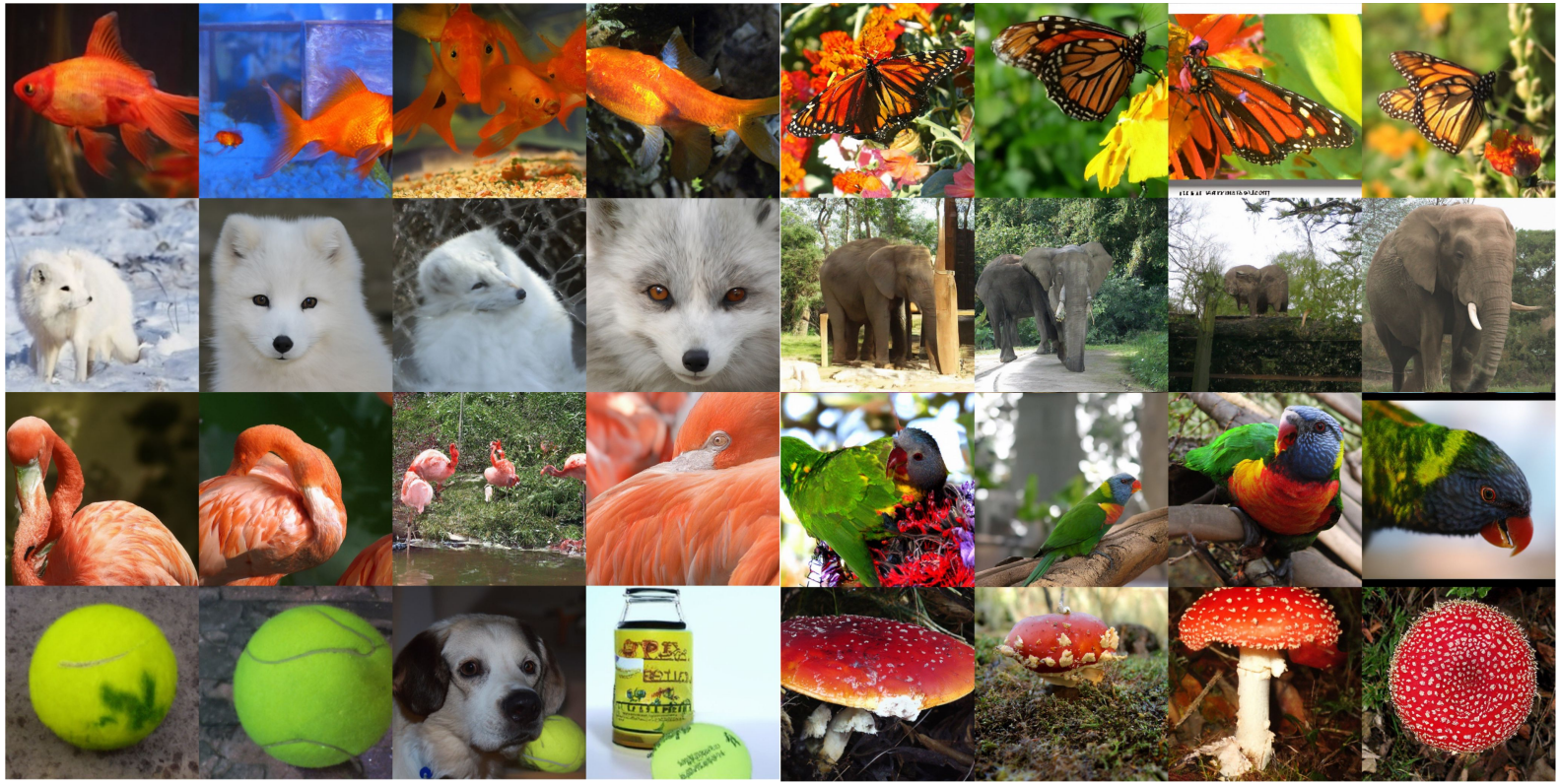


StyleGAN2

ADM-G

Results of ADM(2021)

Some more results.....

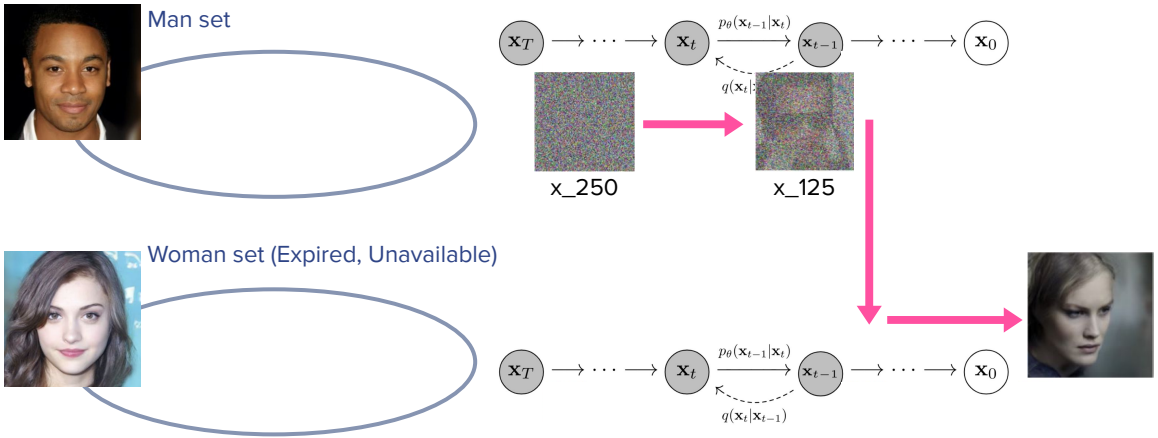


Thoughts

- Shortcoming: Very, very long inference time
  - T=250~1000 reverse process -> Need model forwarding at each timestamp
  - Methods(reduce steps, distill to single model, etc.) were introduced, but not enough yet.
  - How to successfully aggregate some reverse steps?
    - It actually doesn't have to be T steps after the path is found. -> Should check Score-Matching first...😓

Thoughts

- Shortcoming: Very, very long inference time
- Most different point: No Latent space. Generation process.
  - Pros and Cons may co-exist,
  - How about: **1)** For a diffusion models trained in an unreachable domain **2)** Use intermediate state from a diffusion model trained on currently-available data, and pass(or interpolate) it?



Thoughts

- Shortcoming: Very, very long inference time
- Most different point: No Latent space. Generation process.
- Save sampling time with Inference-time Scheduling
  - Actually all T = 1000 in ADM,
  - But did sampling smaller number from t = [0~199, 200~499, 500~699, 700~999] by:

Schedule	FID
50, 50, 50, 50, 50	2.31
70, 60, 50, 40, 30	2.17
90, 50, 40, 40, 30	2.10
90, 60, 50, 30, 20	2.09
80, 60, 50, 30, 30	2.09
90, 50, 50, 30, 30	2.07
100, 50, 40, 30, 30	2.03
90, 60, 60, 20, 20	**2.02**

Rough sweep on LSUN-Bedroom

Schedule	FID	sFID	Prec	Rec
LSUN Bedrooms 256×256				
1000 steps	1.90	5.59	0.66	0.51
250 steps (uniform)	2.31	6.12	0.65	0.50
250 steps (sweep)	2.02	6.12	0.67	0.50
LSUN Horses 256×256				
1000 steps	2.57	6.81	0.71	0.55
250 steps (uniform)	3.45	7.55	0.68	0.56
250 steps (sweep)	2.83	7.08	0.69	0.56
LSUN Cat 256×256				
1000 steps	5.57	6.69	0.63	0.52
250 steps (uniform)	7.03	8.24	0.60	0.53
250 steps (sweep)	5.94	7.43	0.62	0.52

250 steps seems okay, with some

Thanks!

Thanks!

Thanks!

Thanks!

Thanks!

Thanks!

Thanks!

Thanks!