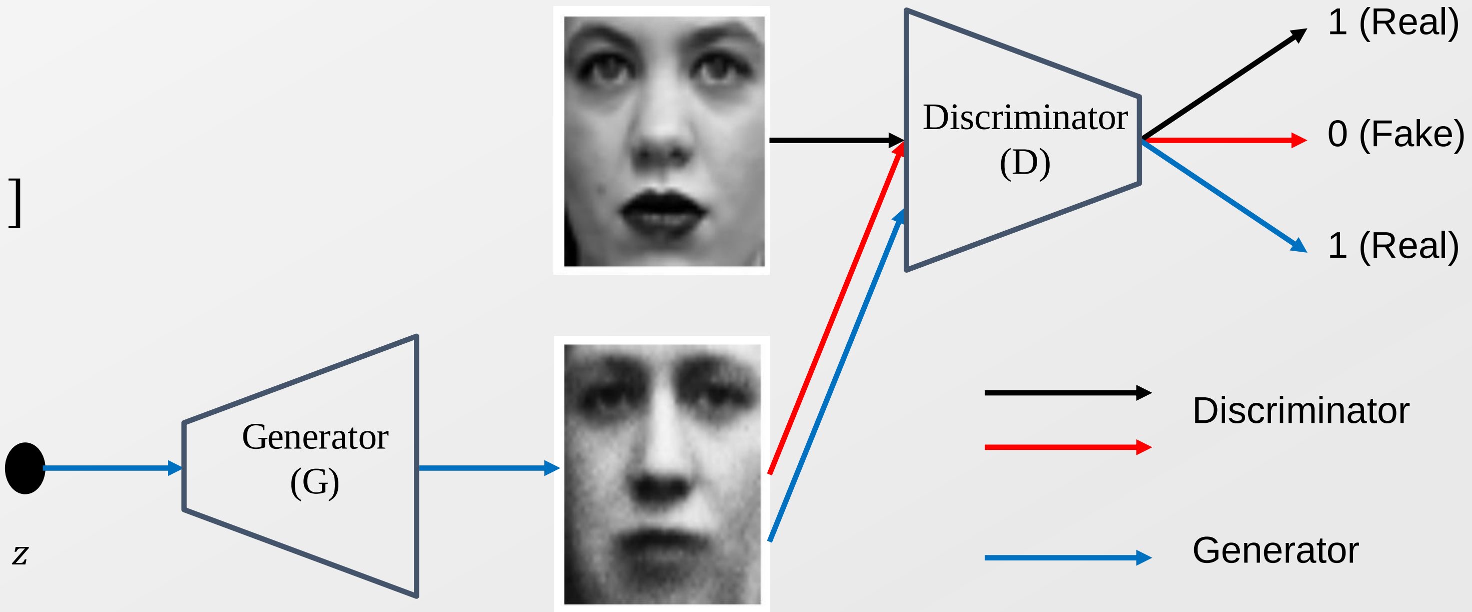
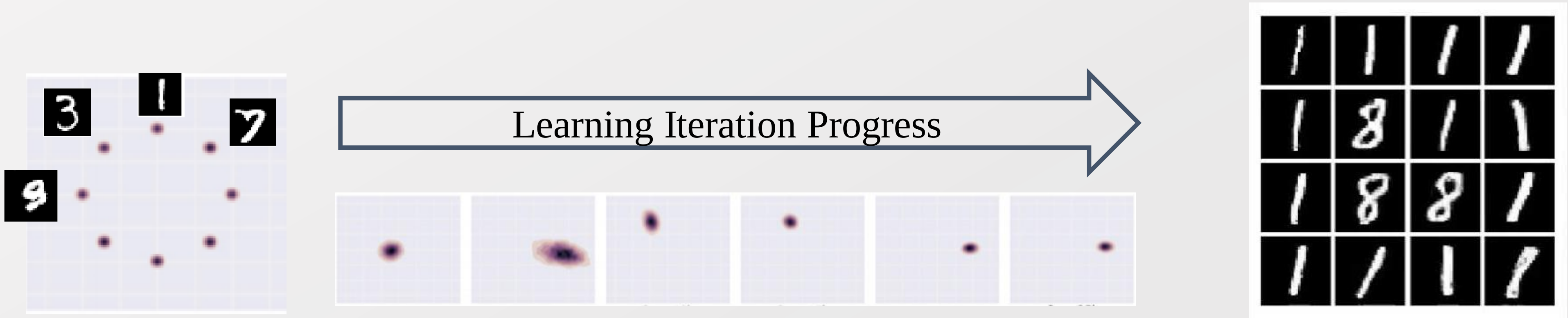
Diffusion Models

Il-Chul Moon
Department of Industrial and Systems Engineering
KAIST
icmoon@kaist.ac.kr

Learning Structure of GAN

- Learning objective function
  - $\min_{G} \max_{D} \; E_{x \sim p_{data(x)}}[logD(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(z)))]$
- Critical Issue of GAN
  - Mode collapse, Non-convergence, Performance degradation



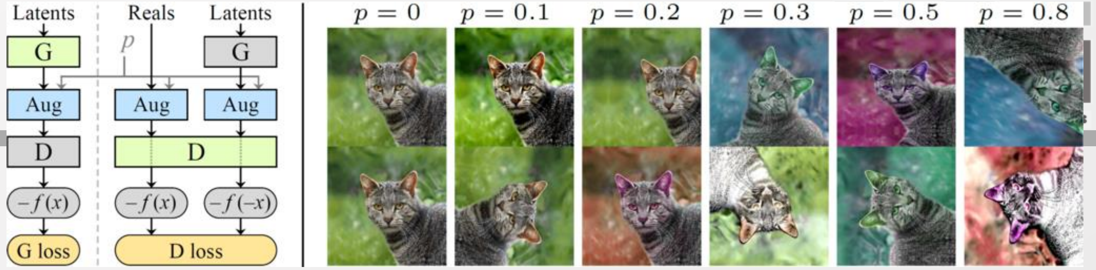


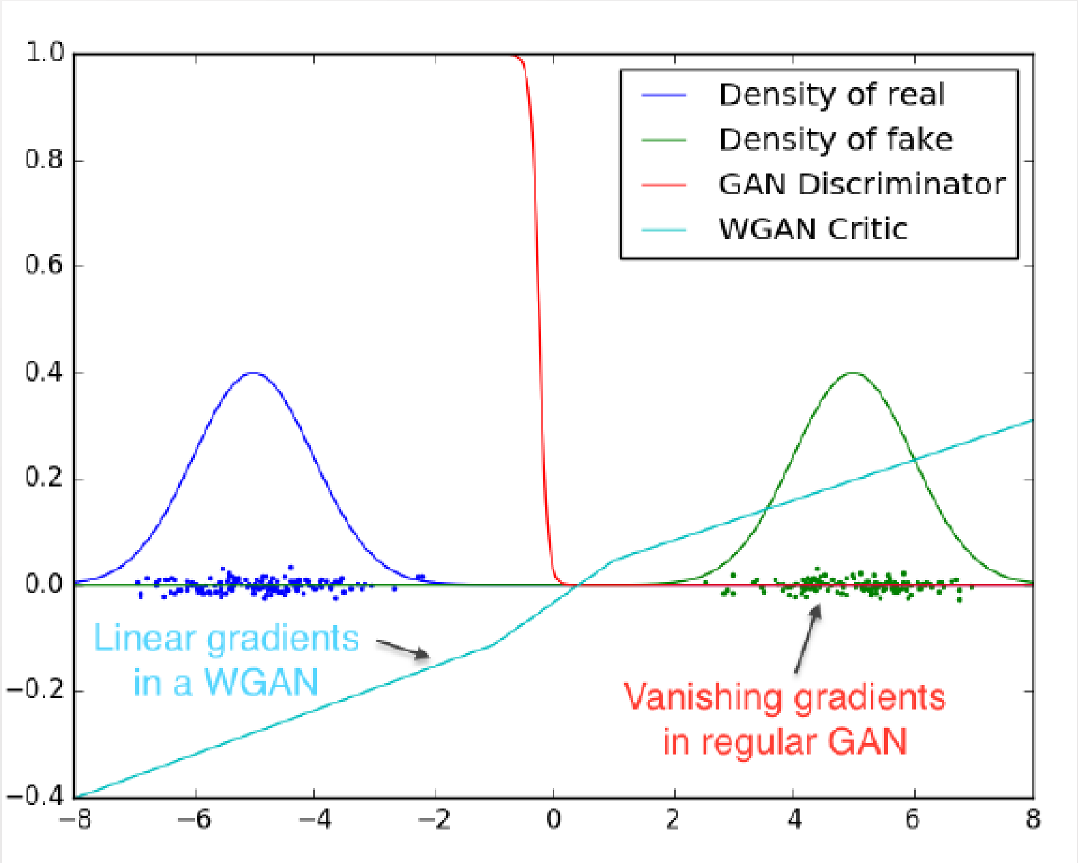
Generates only a subset of modes in p_{data}

Metz, Luke, et al. "Unrolled generative adversarial networks." *arXiv preprint arXiv:1611.02163* (2016).

Learning Divergence of GAN

- Learning objective function
  - $\min_G \max_D E_{x \sim p_{data(x)}}[logD(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(z))\,]$
  - This can be generalized by f-divergence
    - $D_f(P||Q) = \int_x q(x) f\left(\frac{p(x)}{q(x)}\right) dx$
- Is it related to the critical Issue of GAN?
  - Yes. See the domain of f is $\frac{p(x)}{q(x)}$
  - "No-man's land" between the generated and the data instances?
- Method 1. Change the divergence to something else, i.e. Integral Probability Metrics
- Method 2. Fill the GAP with "augmented" data instances





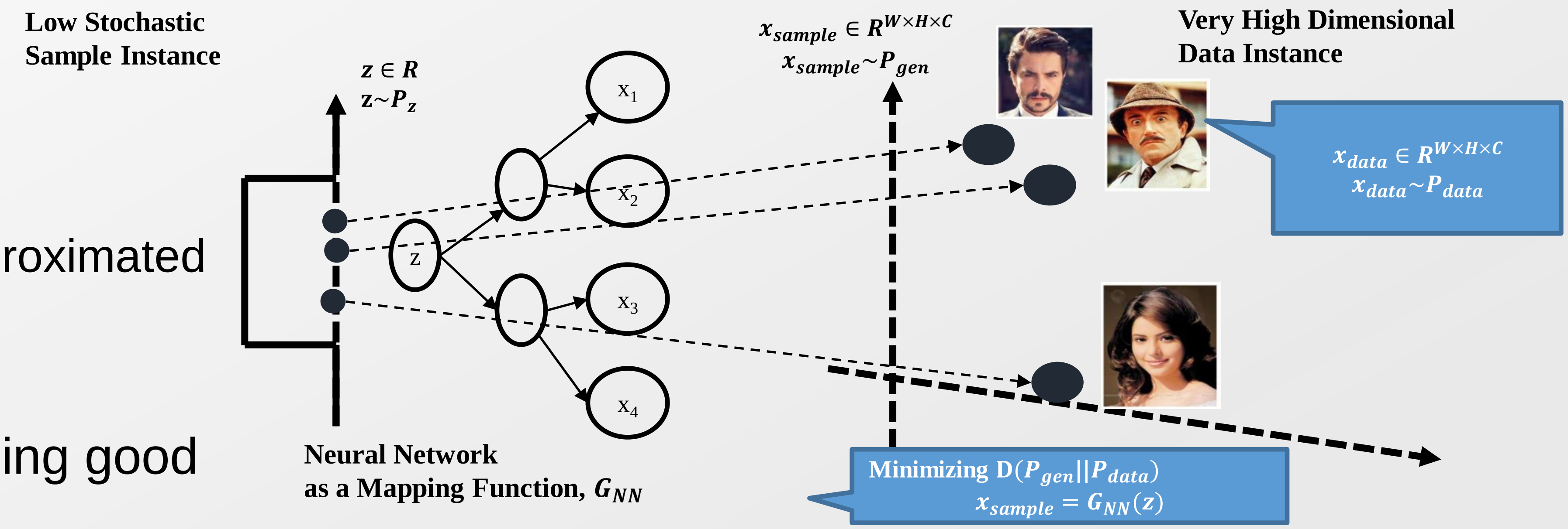
Nowozin, Sebastian, Botond Cseke, and Ryota Tomioka. "f-gan: Training generative neural samplers using variational divergence minimization." *Proceedings of the 30th International Conference on Neural Information Processing Systems*. 2016.

Karras, Tero, et al. "Training generative adversarial networks with limited data." *arXiv preprint arXiv:2006.06676* (2020).

DIFFUSION MODEL

How to Map Stochastic Latent Space to Data Space?

- GAN maps a stochastic sample to a data instance
  - Through Generator function approximated as a neural network

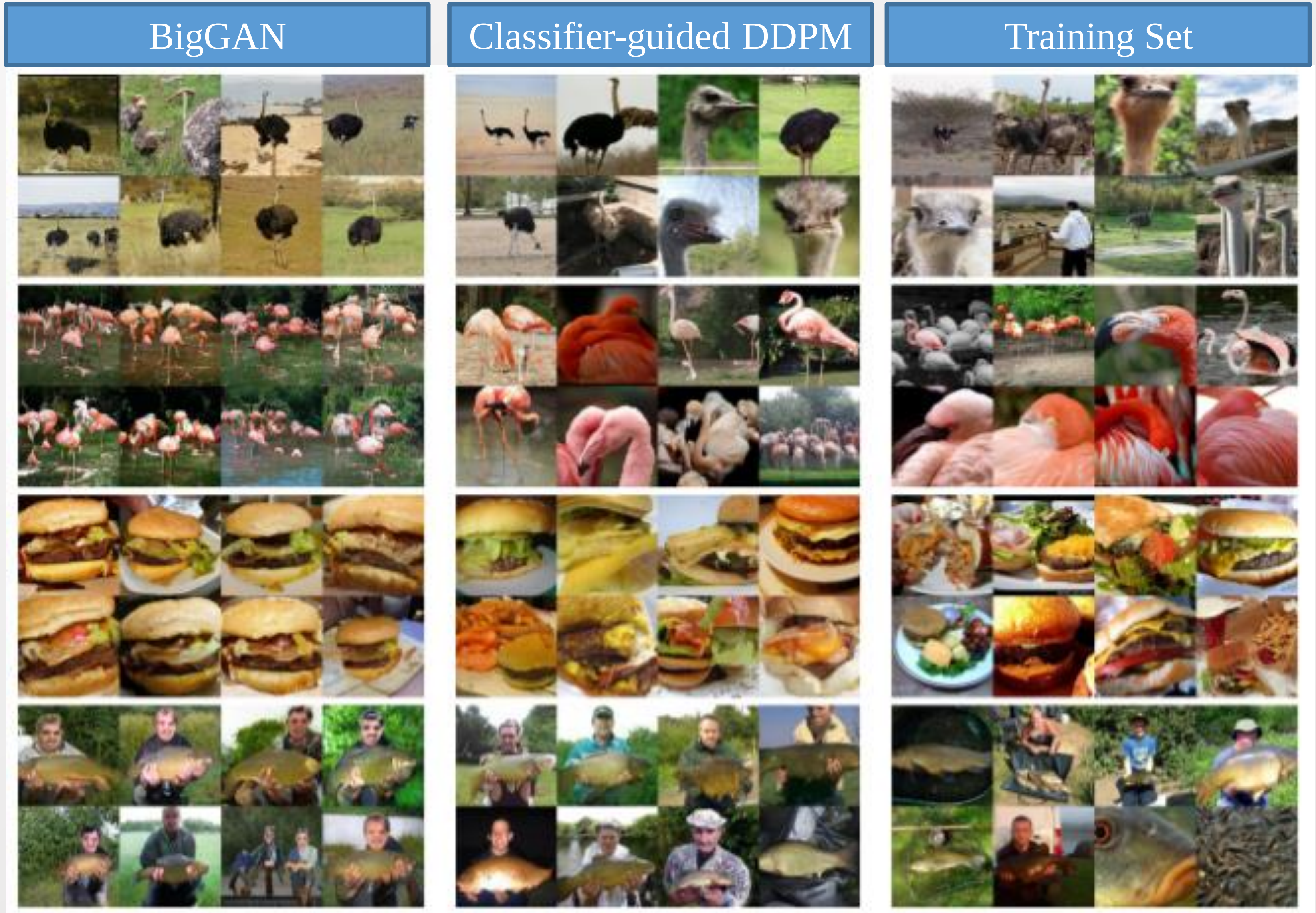


- Is the single-shot mapping good enough?
  - It could have been good enough
    - If the function is flexible enough
      - But, Training of generator → MinMax problem → Inconsistent gradient signal
    - If the mapping of the stochastic element covers the whole data space
      - But, Exploration/exploitation → Mode collapse

- Is there any way to avoid the short-comings of the above?

Diffusion Models Beat GANs on Image Synthesis

- Sampling performances
  - BigGAN-deep
    - With truncation 1.0
    - FID 6.95
  - DDPM
    - With guidance
    - FID 4.59
- Apparent mode collapse from GAN
  - Whereas, DDPM shows no mode collapse
- Is adversarial model better in sampling?
  - Which it can only do…

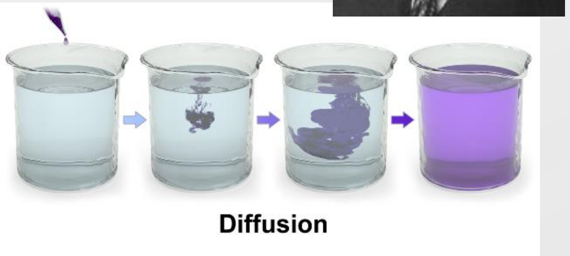


Dhariwal, Prafulla, and Alex Nichol. "Diffusion models beat gans on image synthesis." *arXiv preprint arXiv:2105.05233* (2021).

Diffusion

“The law that entropy always increases holds, I think…”
- Arthur Edington

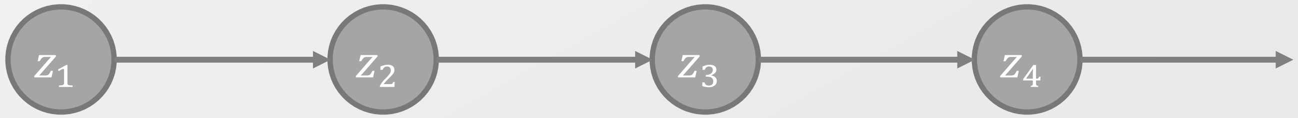
- Flow of particles from high-density regions towards low-density regions
  - Eventually, creating the high entropy in the particle distribution
- Considering a probability distribution over the data space, P_d on R^d
  - P_d : current complex distribution over the data space
  - P_0 : prior distribution over the data space
  - There will be a transformation in the particle distribution over the same space
    - $T: R^d \rightarrow R^d$
    - $x_0 \sim P_d$
    - $T(x_0) \sim P_0$
- We know such techniques of gradual transformation from one distribution to another
  - Markov chain Monte-Carlo
  - Each transition changes from an arbitrary distribution to a stationary distribution
  - Basics of sampling-based inference
    - Metropolis-Hastings algorithm
    - Gibbs sampling



© wikipedia

Detour) Markov Chain for Sampling

- Problem of the previous samplings?
  - No use of the past records → every sampling is independent
- Assigning Z values is a key in the inference
  - Let's assign the values by sampling result
    - Calculate P(E|MC=T,A=F) → Toss a biased coin to assign a value to E
- Sequence of random variables such a process moves through, with the Markov property defining serial dependence only between adjacent periods (as in a "chain")
- A Markov chain is a stochastic process with the Markov property
  - Example) First-order Markov chain



    - $p\left(\mathbf{z}^{(m+1)} \mid \mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\right) = p\left(\mathbf{z}^{(m+1)} \mid \mathbf{z}^{(m)}\right), m \in \{1, \dots, M-1\}$
- Describing systems that follow a chain of linked events, where what happens next depends only on the current state of the system

Detour) Markov chain theory vs. Markov Chain Monte Carlo

APPLIED ARTIFICIAL INTELLIGENCE LAB

- Traditional Markov Chain analysis :
  - A transition rule, $p\left(z^{(t+1)} \mid z^{(t)}\right)$, is given,
  - Interested in finding the stationary distribution $\pi(z)$

$$z_1 \to z_2 \to z_3 \to z_4 \to$$

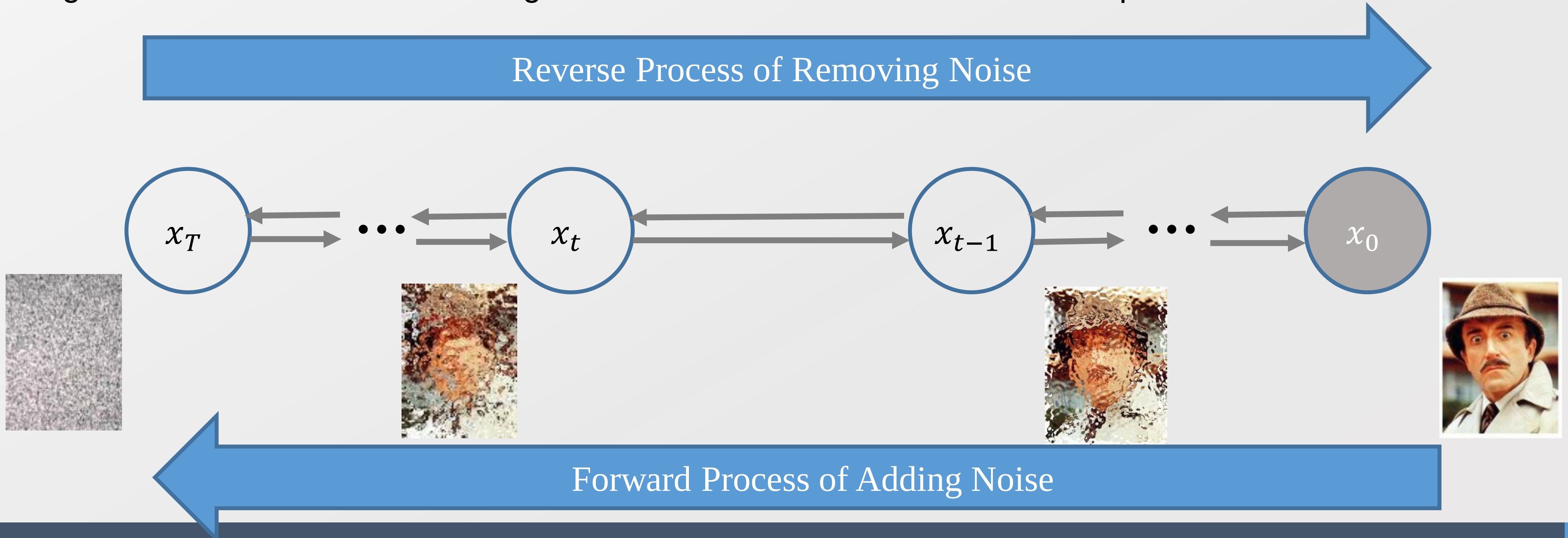
- Markov chain Monte Carlo(MCMC) :
  - A target stationary distribution $\pi(z)$ is known,
  - Interested in prescribing an efficient transition rule to reach the stationary distribution
  - Algorithms for sampling from probability distributions based on constructing a Markov chain that has the desired distribution $\pi(z)$
  - Starting from an arbitrary state, the Markov chain proceeds

$$\underbrace{z^{(1)} \to z^{(2)} \to \cdots \to z^{(m)}}_{Burn-in\ period} \to \underbrace{z^{(m+1)} \to z^{(m+2)} \to \cdots \to z^{(m+n)}}_{Treat\ them\ as\ samples\ from\ \pi(x)}$$

KAIST

Diffusion Process

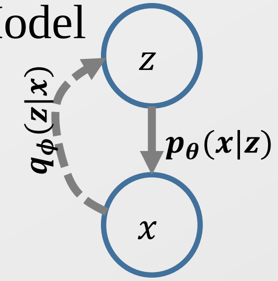
- Let's take an image
  - Forward Process : Gradually add noise, so the image becomes a simple noise
  - Reverse Process : Gradually remove noise, so the noise becomes the image
- This is still a mapping function from the latent space of the noise to the data space
  - There is a single approximated function, the denoised function (Assuming adding a noise becomes trivial)
  - There is no game of generator vs. discriminator
    - The gradient signal is not dis-aligned
    - The generator is not fixated to a single mode to induce the discriminator output.

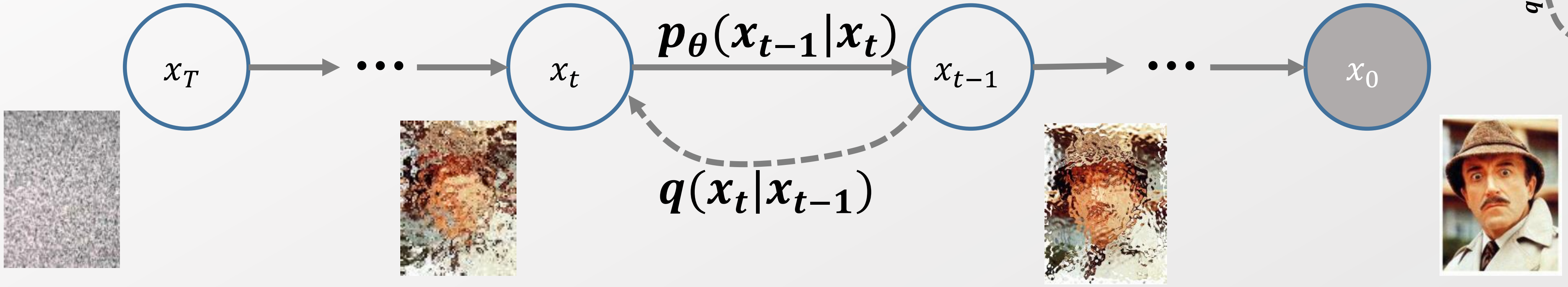


Diffusion Model (Score-Based Model)

- Generative model + Neural network
  - Neural network is used for the inference on the diffusion process

Graphical Model of VAE





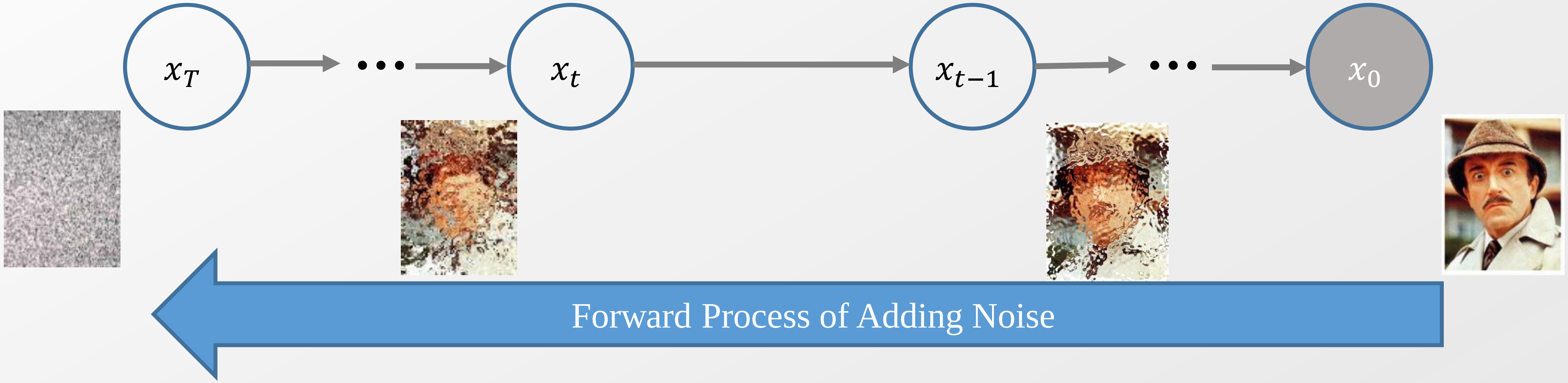
  - Basically, a Markov chain by shaping an instance, or destroying the structure on an instance

$$p_\theta(x_{0:T}) = p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)\,, p_\theta(x_{t-1}|x_t) = N(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})\,, q(x_t|x_{t-1}) = N(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

  - Similar to VAEs, given the neural-network based inference on the distribution parameters
    - Not exactly the same, VAE does not add potential noises to an instance over time, or over layers
    - Some group of VAEs with hierarchical z becomes similar to the diffusion model

Forward Diffusion Process of Adding Noise



- Stochastic mapping from the data space to the latent space
  - Can be regarded as an encoding process
  - Instead, this is going to be a fixed encoder without learning
- As a single-step adding noise,
  - $q(x_t|x_{t-1}) = N(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$
  - Actual sampling could be re-parametrized as
    - $x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon_{t-1},\ \epsilon_{t-1} \sim N(0, I)$
- This single-step noise addition will take a long-time if $T \rightarrow \infty$,
  - So, we need a solution of this stochastic process, which is $q(x_t|x_0)$

Forward Diffusion with Single Computation

Help from Dongjun Kim

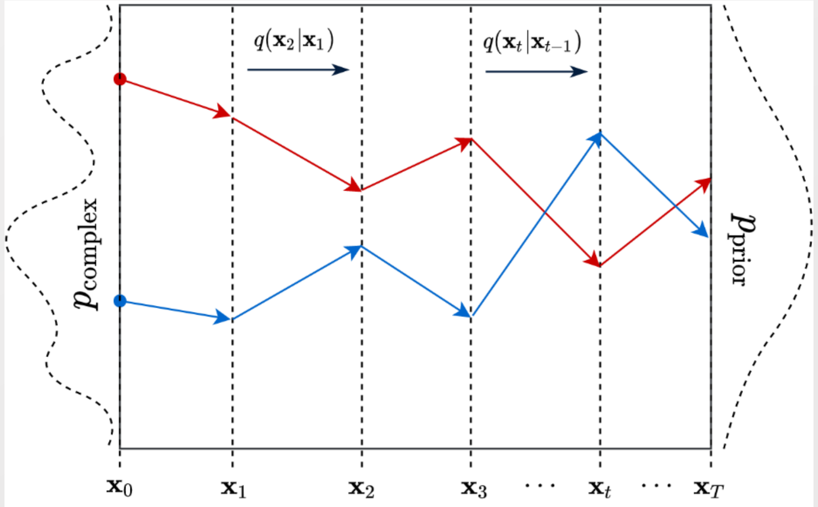
- $q(x_t|x_{t-1}) = N\left(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I\right)$
- Generalizing $q(x_t|x_0)$ by mathematical induction
  - $q(x_1|x_0) = N(x_1; \sqrt{1-\beta_1}x_0, \beta_1 I)$
- Let's assume $q(x_{t-1}|x_0) = N\left(x_{t-1}; \sqrt{\prod_{s=1}^{t-1}(1-\beta_s)}x_0, (1-\prod_{s=1}^{t-1}(1-\beta_s))I\right)$
- Assuming x_{t-1} and ϵ_{t-1} follow the Gaussian distribution
  - $x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon_{t-1}$, $\epsilon_{t-1} \sim N(0, I)$
  - Sum of Gaussian distributions are still Gaussian distribution
    - $X \sim N(\mu_X, \sigma_X^2)$, $Y \sim N(\mu_Y, \sigma_Y^2)$, $Z = X + Y$, $Z \sim N(\mu_X + \mu_Y, \sigma_X^2 + \sigma_Y^2)$
- Let's say
  - $X = \sqrt{1-\beta_t}x_{t-1} \sim N\left(\sqrt{1-\beta_t}\sqrt{\prod_{s=1}^{t-1}(1-\beta_s)}x_0, \left(\sqrt{1-\beta_t}\right)^2(1-\prod_{s=1}^{t-1}(1-\beta_s))I\right)$
  - $Y = \sqrt{\beta_t}\epsilon_{t-1} \sim N\left(0, \left(\sqrt{\beta_t}\right)^2 I\right)$
- Then, $x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon_{t-1}$
- $= X + Y \sim N\left(\sqrt{\prod_{s=1}^{t}(1-\beta_s)}x_0, (1-\beta_t)(1-\prod_{s=1}^{t-1}(1-\beta_s))I + \beta_t I\right)$

$$= N\left(\sqrt{\prod_{s=1}^{t}(1-\beta_s)}x_0, \left(1-\prod_{s=1}^{t}(1-\beta_s)\right)I\right)$$

$$(1-\beta_t)\left(1-\prod_{s=1}^{t-1}(1-\beta_s)\right)+\beta_t$$
$$= 1-\prod_{s=1}^{t-1}(1-\beta_s) - \beta_t + \beta_t\prod_{s=1}^{t-1}(1-\beta_s) + \beta_t$$
$$= 1-\prod_{s=1}^{t-1}(1-\beta_s) + \beta_t\prod_{s=1}^{t-1}(1-\beta_s)$$
$$= 1-(1-\beta_t)\prod_{s=1}^{t-1}(1-\beta_s)$$
$$= 1-\prod_{s=1}^{t}(1-\beta_s)$$

Diffusion Kernel

- $q(x_t|x_0) = N\left(\sqrt{\prod_{s=1}^{t}(1-\beta_s)}x_0, (1-\prod_{s=1}^{t}(1-\beta_s))I\right) = N(\sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I)$
  - $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$
  - $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{(1-\bar{\alpha}_t)}\epsilon,\ \epsilon \sim N(0, I)$
- $\bar{\alpha}_T \to 0 \Rightarrow q(x_T|x_0) \approx N(0, I)$: This requires a schedule on β_t
- Effect of the forward diffusion from the diffusion kernel perspective
  - $q(x_t) = \int q(x_0, x_t)dx_0 = \int q(x_t|x_0)q(x_0)dx_0$
  - Here, $q(x_t|x_0)$ becomes the kernel of the Gaussian convolution.
  - This shows the sampling of $x_t \sim q(x_t|x_0)$
    - Sample $x_0 \sim N(0, I)$ → Sample $x_t \sim q(x_t|x_0)$: a.k.a. ancestral sampling

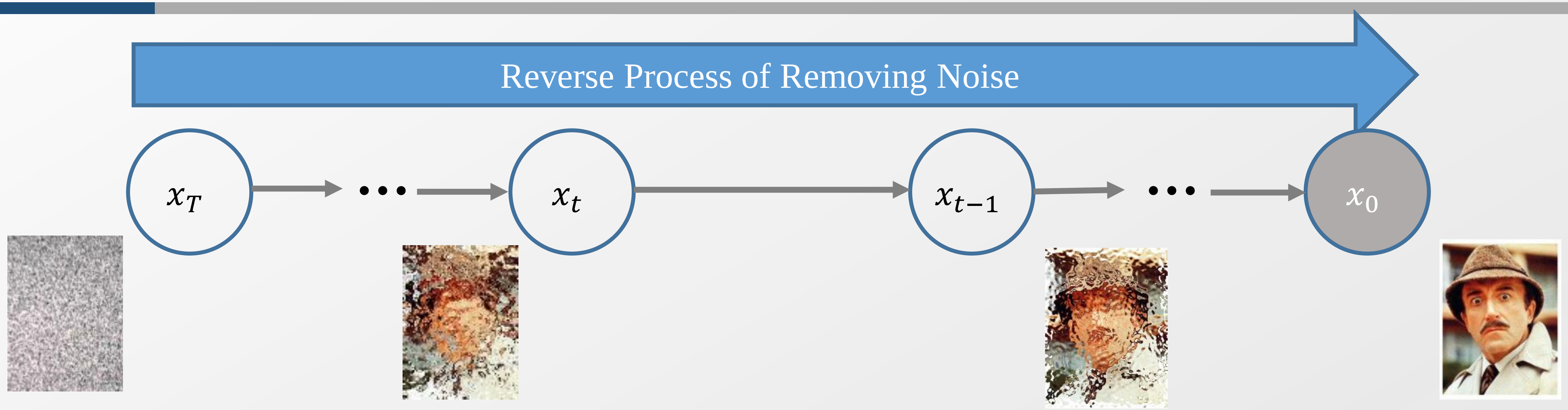


$x_T \to \cdots \to x_t \to x_{t-1} \to \cdots \to x_0$




Forward Process of Adding Noise

Reverse Diffusion as Denoising Process



- Forward diffusion provides no real merit in the generation task
  - On the opposite, reverse diffusion will directly become the generation of samples from $N(0, I)$
  - $x_T \sim N(0, I)$
  - $x_{t-1} \sim p(x_{t-1}|x_t)$: True denoising distribution
    - In simple approach : $p(x_{t-1}|x_t) \propto q(x_{t-1})q(x_t|x_{t-1})$ → This becomes intractable in sampling
    - Then, we need a direct approximation on $p(x_{t-1}|x_t)$
      - By the flexibility and complexity of neural networks
- $p_\theta(x_{t-1}|x_t) = N(\mu_\theta(x_t, t), \sigma_t^2 I)$, $p(X_T) = N(0, I)$
  - Fixed covariance structure and only trainable function of $\mu_\theta(x_t, t)$

Transition Probabilities in Summary

- Our goal becomes the training of $p_\theta(x_{t-1}|x_t) = N(\mu_\theta(x_t, t), \sigma_t^2 I)$
- Therefore, we need to compare the distribution between $p_\theta(x_{t-1}|x_t)$ and $q(x_t|x_{t-1})$ given x_0
  - $p_\theta(x_{t-1}|x_t) = N(\mu_\theta(x_t, t), \sigma_t^2 I)$
  - $q(x_t|x_{t-1}) = N\left(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I\right)$
    - $q(x_t|x_0) = N\left(\sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I\right)$
      - $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s) = \alpha_t\bar{\alpha}_{t-1}$
      - $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{(1-\bar{\alpha}_t)}\epsilon,\ \epsilon \sim N(0, I)$
      - $x_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \sqrt{(1-\bar{\alpha}_t)}\epsilon\right),\ \epsilon \sim N(0, I)$
    - $q(x_{t-1}|x_t, x_0) = N(\tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I)$
      - The convolution of x_t and x_0
      - $\tilde{\mu}_t(x_t, x_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t$
      - $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$
- The match becomes $q(x_{t-1}|x_t, x_0)$ and $p_\theta(x_{t-1}|x_t)$

Loss Structure from Reverse Diffusion (1)

- Our goal becomes the training of $p_\theta(x_{t-1}|x_t) = N(\mu_\theta(x_t,t),\sigma_t^2 I)$
- Therefore, we need to compare the distribution between $p_\theta(x_{t-1}|x_t)$ and $q(x_t|x_{t-1})$
  - $p_\theta(x_{t-1}|x_t) = N(\mu_\theta(x_t,t),\sigma_t^2 I)$
  - $q(x_{t-1}|x_t,x_0) = N\left(\tilde{\mu}_t(x_t,x_0),\tilde{\beta}_t I\right)$

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

$$\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

$$\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s) = (1-\beta_t)\bar{\alpha}_{t-1} = \alpha_t\bar{\alpha}_{t-1}$$

- Let's match the mean function $\mu_\theta(x_t,t)$ to $\tilde{\mu}_t(x_t,x_0)$ in $N(\mu_\theta(x_t,t),\sigma_t^2 I)$
  - $\sigma_t^2 = \tilde{\beta}_t$
  - $L_{t-1} = E_{x_0\sim p(x_0),\epsilon\sim N(0,I)}\left[\frac{1}{2\sigma_t^2}\left|\left|\tilde{\mu}_t(x_t,x_0) - \mu_\theta(x_t,t)\right|\right|^2\right]$

$$x_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \sqrt{(1-\bar{\alpha}_t)}\epsilon\right)$$

$$= E_{x_0\sim p(x_0),\epsilon\sim N(0,I)}\left[\frac{1}{2\sigma_t^2}\left\|\tilde{\mu}_t\left(x_t,\frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t-\sqrt{(1-\bar{\alpha}_t)}\epsilon\right)\right)-\mu_\theta(x_t,t)\right\|^2\right]$$

- Then, we need to take the inputs of $\tilde{\mu}_t$ to compare μ_θ
  - How to simplify $\tilde{\mu}_t\left(x_t,\frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t-\sqrt{(1-\bar{\alpha}_t)}\epsilon\right)\right)$?

Loss Structure from Reverse Diffusion (2)

- $\tilde{\mu}_t(x_t, x_0) = \tilde{\mu}_t\left(x_t, \frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \sqrt{(1-\bar{\alpha}_t)}\epsilon\right)\right)$

$$= \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\left(\frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \sqrt{(1-\bar{\alpha}_t)}\epsilon\right)\right) + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t$$

$$= \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\frac{1}{\sqrt{\bar{\alpha}_t}}x_t + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t - \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\frac{\sqrt{(1-\bar{\alpha}_t)}}{\sqrt{\bar{\alpha}_t}}\epsilon$$

$$= \frac{1}{(1-\bar{\alpha}_t)\sqrt{\bar{\alpha}_t}}\left(\sqrt{\bar{\alpha}_{t-1}}\beta_t + \sqrt{\bar{\alpha}_t}\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\right)x_t - \frac{\sqrt{\bar{\alpha}_{t-1}}}{\sqrt{(1-\bar{\alpha}_t)}\sqrt{\bar{\alpha}_t}}\beta_t\epsilon$$

$$= \frac{1}{(1-\bar{\alpha}_t)\sqrt{\bar{\alpha}_t}}\left(\sqrt{\bar{\alpha}_{t-1}}\beta_t + \sqrt{\bar{\alpha}_{t-1}}\sqrt{\alpha_t}\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\right)x_t - \frac{\sqrt{\bar{\alpha}_{t-1}}}{\sqrt{(1-\bar{\alpha}_t)}\sqrt{\alpha_t\bar{\alpha}_{t-1}}}\beta_t\epsilon$$

$$= \frac{\sqrt{\bar{\alpha}_{t-1}}}{(1-\bar{\alpha}_t)\sqrt{\alpha_t\bar{\alpha}_{t-1}}}\left(\beta_t + \alpha_t(1-\bar{\alpha}_{t-1})\right)x_t - \frac{\beta_t}{\sqrt{(1-\bar{\alpha}_t)}\sqrt{\alpha_t}}\epsilon$$

$$= \frac{1}{(1-\bar{\alpha}_t)\sqrt{\alpha_t}}\left(1-\alpha_t + \alpha_t(1-\bar{\alpha}_{t-1})\right)x_t - \frac{\beta_t}{\sqrt{(1-\bar{\alpha}_t)}\sqrt{\alpha_t}}\epsilon$$

$$= \frac{1}{(1-\alpha_t\bar{\alpha}_{t-1})\sqrt{\alpha_t}}(1-\alpha_t\bar{\alpha}_{t-1})x_t - \frac{\beta_t}{\sqrt{(1-\bar{\alpha}_t)}\sqrt{\alpha_t}}\epsilon = \frac{1}{\sqrt{\alpha_t}}x_t - \frac{\beta_t}{\sqrt{(1-\bar{\alpha}_t)}\sqrt{\alpha_t}}\epsilon = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{(1-\bar{\alpha}_t)}}\epsilon\right)$$

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

$$x_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \sqrt{(1-\bar{\alpha}_t)}\epsilon\right)$$

$$\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s) = \alpha_t\bar{\alpha}_{t-1}$$

Loss Structure from Reverse Diffusion (3)

- $L_{t-1} = E_{x_0\sim p(x_0),\epsilon\sim N(0,I)}\left[\frac{1}{2\sigma_t^2}||\tilde{\mu}_t(x_t,x_0)-\mu_\theta(x_t,t)||^2\right]$

$$= E_{x_0\sim p(x_0),\epsilon\sim N(0,I)}\left[\frac{1}{2\sigma_t^2}\left\|\tilde{\mu}_t\left(x_t,\frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t-\sqrt{(1-\bar{\alpha}_t)}\epsilon\right)\right)-\mu_\theta(x_t,t)\right\|^2\right]$$

$$= E_{x_0\sim p(x_0),\epsilon\sim N(0,I)}\left[\frac{1}{2\sigma_t^2}\left\|\frac{1}{\sqrt{\alpha_t}}\left(x_t-\frac{\beta_t}{\sqrt{(1-\bar{\alpha}_t)}}\epsilon\right)-\mu_\theta(x_t,t)\right\|^2\right]$$

$$= E_{x_0\sim p(x_0),\epsilon\sim N(0,I)}\left[\frac{1}{2\sigma_t^2}\left\|\frac{1}{\sqrt{\alpha_t}}\left(x_t(x_0,\epsilon)-\frac{\beta_t}{\sqrt{(1-\bar{\alpha}_t)}}\epsilon\right)-\frac{1}{\sqrt{\alpha_t}}\left(x_t(x_0,\epsilon)-\frac{\beta_t}{\sqrt{(1-\bar{\alpha}_t)}}\epsilon_\theta(x_t,t)\right)\right\|^2\right]$$

$$= E_{x_0\sim p(x_0),\epsilon\sim N(0,I)}\left[\frac{1}{2\sigma_t^2}\left\|\frac{1}{\sqrt{\alpha_t}}\left(-\frac{\beta_t}{\sqrt{(1-\bar{\alpha}_t)}}\epsilon+-\frac{\beta_t}{\sqrt{(1-\bar{\alpha}_t)}}\epsilon_\theta(x_t,t)\right)\right\|^2\right]$$

$$= E_{x_0\sim p(x_0),\epsilon\sim N(0,I)}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}||\epsilon-\epsilon_\theta(x_t,t)||^2\right]$$

$$= E_{x_0\sim p(x_0),\epsilon\sim N(0,I)}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}\left\|\epsilon-\epsilon_\theta\left(\sqrt{\bar{\alpha}_t}x_0+\sqrt{(1-\bar{\alpha}_t)}\epsilon,t\right)\right\|^2\right]$$

We assume the parameterization of

$$\mu_\theta(x_t,t)=\tilde{\mu}_t\left(x_t,\frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t-\sqrt{(1-\bar{\alpha}_t)}\epsilon_\theta(x_t)\right)\right)$$

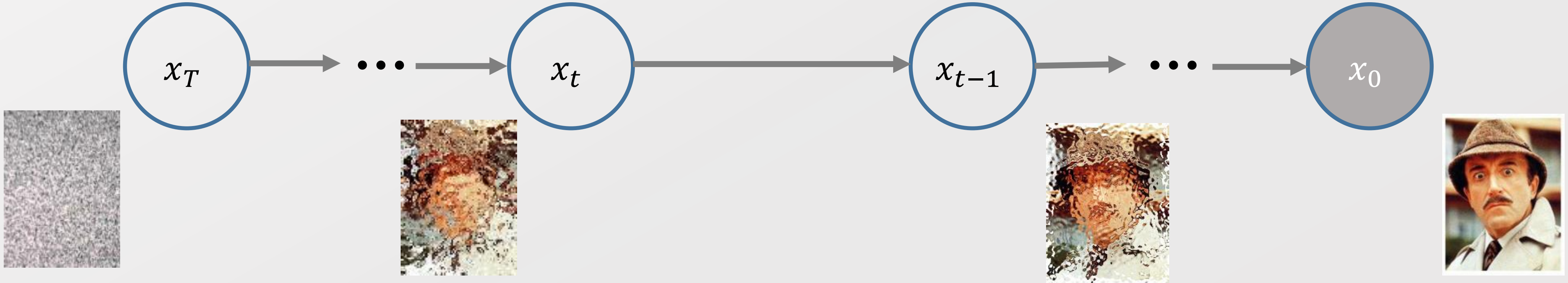
$$=\frac{1}{\sqrt{\alpha_t}}\left(x_t-\frac{\beta_t}{\sqrt{(1-\bar{\alpha}_t)}}\epsilon_\theta(x_t,t)\right)$$

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

Loss Structure from Entire Reverse Diffusion (1)

- Loss structure for a single step of Markov chain
  - x_0 is the only observed variable, and $x_T \dots x_1$ are all latent variables
  - Since they are latent variables, there should be an ELBO structure for the MLE learning
    - To match the loss direction as a minimization and the ELBO direction as a maximization; we will use the negative ELBO to minimize “NELBO”.
- $E[-\log p_\theta(x_0)] \le E_q\left[-\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\right] = E_q\left[-\log \frac{p(x_T)\prod_{1\le t\le T} p_\theta(x_{t-1}|x_t)}{\prod_{1\le t\le T} q(x_t|x_{t-1})}\right]$

$$= E_q\left[-\log p(x_T) - \sum_{1\le t\le T} \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}\right] = L$$



Loss Structure from Entire Reverse Diffusion (2)

- $L = E_q\left[-\log p(x_T) - \sum_{1\le t\le T}\log\frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}\right]$

$$= E_q\left[-\log p(x_T) - \sum_{1<t\le T}\log\frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})} - \log\frac{p_\theta(x_0|x_1)}{q(x_1|x_0)}\right]$$

$$= E_q\left[-\log p(x_T) - \sum_{1<t\le T}\log\frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t,x_0)}\frac{q(x_{t-1}|x_0)}{q(x_t|x_0)} - \log\frac{p_\theta(x_0|x_1)}{q(x_1|x_0)}\right]$$

$$= E_q\left[-\log\frac{p(x_T)}{q(x_T|x_0)} - \sum_{1<t\le T}\log\frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t,x_0)} - \log p_\theta(x_0|x_1)\right]$$

$$= E_q\left[D_{KL}\big(q(x_T|x_0)\,||\,p(x_T)\big) + \sum_{1<t\le T} D_{KL}\big(q(x_{t-1}|x_t,x_0)\,||\,p_\theta(x_{t-1}|x_t)\big) - \log p_\theta(x_0|x_1)\right]$$

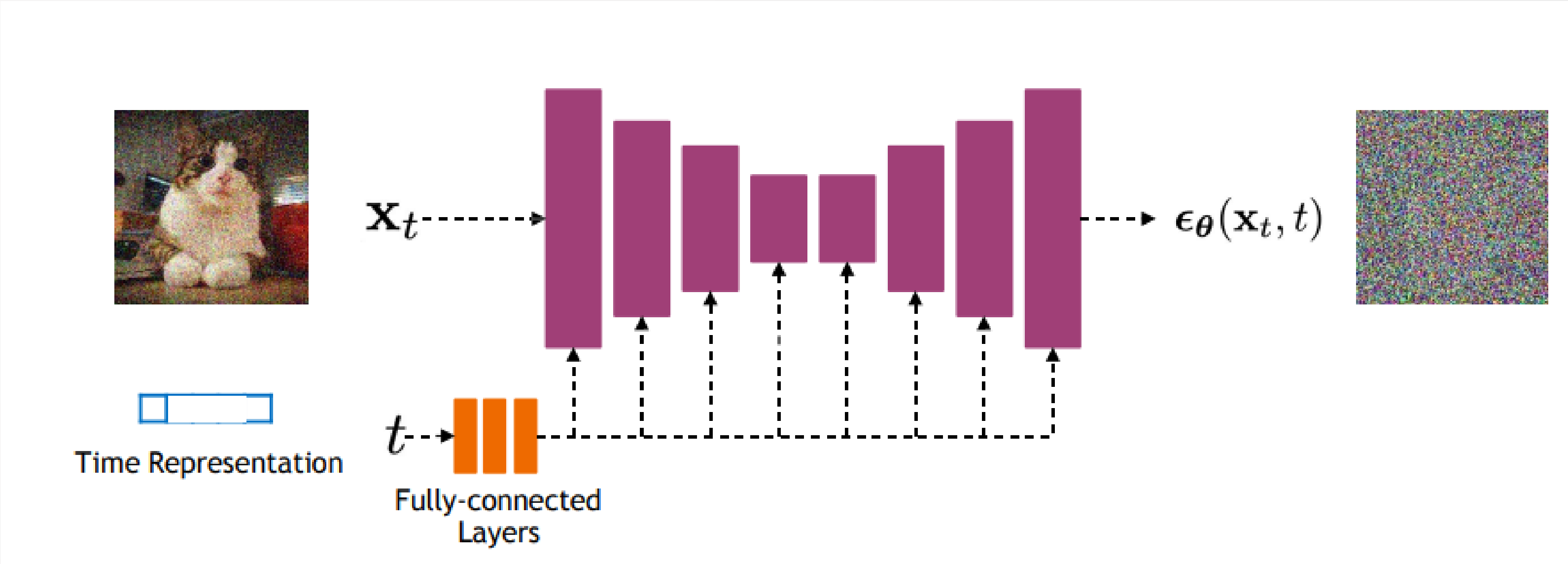
- L_{t-1} matches two distributions of $p_\theta(x_{t-1}|x_t) = N(\mu_\theta(x_t,t),\sigma_t^2 I)$ and $q(x_{t-1}|x_t,x_0) = N\big(\tilde{\mu}_t(x_t,x_0),\tilde{\beta}_t I\big)$
  - Which consistutes the second term of the entire loss.

Actual Implementation of Noise Pattern Prediction

- A single step of diffusion model constitutes
  - $L_{t-1} = E_{x_0 \sim p(x_0), \epsilon \sim N(0,I)} \left[\frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1-\bar{\alpha}_t)} \left|\left| \epsilon - \epsilon_\theta(x_t, t) \right|\right|^2 \right]$

$$= E_{x_0 \sim p(x_0), \epsilon \sim N(0,I)} \left[\frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1-\bar{\alpha}_t)} \left|\left| \epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{(1-\bar{\alpha}_t)}\epsilon, t\right) \right|\right|^2 \right]$$

- What you need is a U-net shaped neural network with inputs of x_t and t
  - x_t can be produced from x_0 in the closed form



Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \ldots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\boxed{\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon},}\ t\right) \right\|^2$$
6: **until** converged

Actual Sampling Procedure of Diffusion Model

- We only modeled the distribution of
  - $p_\theta(x_{t-1}|x_t) = N(\mu_\theta(x_t, t), \sigma_t^2 I)$
  - $q(x_{t-1}|x_t, x_0) = N\left(\tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I\right)$
- Therefore, there is no jump in the decoding step of the series of Markov chain
  - You will utilize the pattern prediction network to denoise the pattern
  - Step-by-Step with long time

Algorithm 2 Sampling

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

Some Effects of Derivation

- $L = E_q\left[-\log p(x_T) - \sum_{1\le t\le T}\log\frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}\right] = E_q\left[-\log\frac{p(x_T)}{q(x_T|x_0)} - \sum_{1<t\le T}\log\frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t,x_0)} - \log p_\theta(x_0|x_1)\right]$
  - $q(x_t|x_{t-1})$ vs. $q(x_{t-1}|x_t,x_0)$
    - Variance reduction effect : x_0 is added to reduce the variance by always providing the grounding evidence in the variational distribution
    - $q(x_{t-1}|x_t,x_0) = N(\tilde{\mu}_t(x_t,x_0),\tilde{\beta}_t I)$ provides the closed form solution
      - $\tilde{\mu}_t(x_t,x_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t$
      - $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$
- $L_{t-1} = E_{x_0\sim p(x_0),\epsilon\sim N(0,I)}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}\left|\left|\epsilon-\epsilon_\theta(x_t,t)\right|\right|^2\right]$
  - $L \approx E_{t\sim Unif(0,T)}\left[E_{x_0\sim p(x_0),\epsilon\sim N(0,I)}\left[\left|\left|\epsilon-\epsilon_\theta(x_t,t)\right|\right|^2\right]\right] \approx \frac{1}{T}\sum_{1\le t<T}p(t)E_{x_0\sim p(x_0),\epsilon\sim N(0,I)}\left[\left|\left|\epsilon-\epsilon_\theta(x_t,t)\right|\right|^2\right]$
  - Refelct to the Denoising Score Matching loss : Relation to the Noise Conditioned Score Network (NCSN)
  - $J_D(\theta,\sigma) = E_{p_{data}(\tilde{x},x)}\left[\left|\left|s_\theta(\tilde{x};\sigma) - \nabla_{\tilde{x}}\log p(\tilde{x}|x)\right|\right|^2\right]$

PERSPECTIVE FROM NCSN

Score Matching

- Ususally, inference task requires the optimization on the PDF
  - $p(\xi;\theta) = \frac{1}{Z(\theta)} q(\xi;\theta)$
  - $z(\theta) = \int_{\xi \in R^n} q(\xi;\theta) d\xi$
  - However, the integration of $z(\theta)$ becomes intractable
  - Therefore, there is a demand to estimate the parameter of non-normalized density models
    - Traditional approach would be MCMC, i.e. Metropolis-Hastings, also special case Gibbs sampling
- Let's say that the score function would be
  - $\psi(\xi;\theta) = \begin{pmatrix} \frac{\partial \log p(\xi;\theta)}{\partial \xi_1} \\ \cdots \\ \frac{\partial \log p(\xi;\theta)}{\partial \xi_n} \end{pmatrix} = \begin{pmatrix} \psi_1(\xi;\theta) \\ \cdots \\ \psi_n(\xi;\theta) \end{pmatrix} = \nabla_\xi \log p(\xi;\theta) = \nabla_\xi \log q(\xi;\theta)$
  - The merit of $\nabla_\xi \log p(\xi;\theta)$ is its independence from $Z(\theta)$
    - From the log-likelihood and the derivative over ξ
- Finally, we define the matching between the model score and the data score functions
  - $J(\theta) = \frac{1}{2} \int_{\xi \in R^n} p_x(\xi) \left|\left|\psi(\xi;\theta) - \psi_x(\xi)\right|\right|^2 d\xi$
  - This directly handles the $q(\xi;\theta)$, but now we need the data score function of $\psi_x(\xi)$

Score Matching from Samples

- $J(\theta) = \frac{1}{2}\int_{\xi \in R^n} p_x(\xi) \left|\left|\psi(\xi;\theta) - \psi_x(\xi)\right|\right|^2 d\xi$
  - This directly handles the $q(\xi;\theta)$, but now we need the data score function of $\psi_x(\xi)$
  - $= \int_{\xi \in R^n} p_x(\xi) \left[\frac{1}{2}\left|\left|\psi(\xi;\theta)\right|\right|^2 + \frac{1}{2}\left|\left|\psi_x(\xi)\right|\right|^2 - \psi_x(\xi)^T\psi(\xi;\theta)\right] d\xi$

$$= \int_{\xi \in R^n} p_x(\xi) \left[\frac{1}{2}\left|\left|\psi(\xi;\theta)\right|\right|^2 - \psi_x(\xi)^T\psi(\xi;\theta)\right] d\xi + C$$

$$= \int_{\xi \in R^n} p_x(\xi) \frac{1}{2}\left|\left|\psi(\xi;\theta)\right|\right|^2 d\xi - \int_{\xi \in R^n} p_x(\xi)\psi_x(\xi)^T\psi(\xi;\theta) d\xi + C$$

$$= \int_{\xi \in R^n} p_x(\xi) \frac{1}{2}\left|\left|\psi(\xi;\theta)\right|\right|^2 d\xi - \sum_i \int_{\xi \in R^n} p_x(\xi)\psi_{x,i}(\xi)\psi_i(\xi;\theta) d\xi + C$$

  - $-\int p_x(\xi)\frac{\partial \log p_x(\xi)}{\partial \xi_i}\psi_i(\xi;\theta)d\xi = -\int p_x(\xi)\frac{\partial p_x(\xi)}{p_x(\xi)\partial \xi_i}\psi_i(\xi;\theta)d\xi = -\int \frac{\partial p_x(\xi)}{\partial \xi_i}\psi_i(\xi;\theta)d\xi$
    - Single dimension case : $\int p(x)(\log p)'(x)f(x) = \int p(x)\frac{p'(x)}{p(x)}f(x)dx = \int p'(x)f(x)dx = -\int p(x)f'(x)dx$
    - Multi dimension case : $-\int \frac{\partial p_x(\xi)}{\partial \xi_i}\psi_{i(\xi;\theta)}d\xi = \int \frac{\partial \psi_{i(\xi;\theta)}}{\partial \xi_i}p_x(\xi)d\xi$
      - Requires some additional proves and assumptions:
        - Hyvärinen, Aapo, and Peter Dayan. "Estimation of non-normalized statistical models by score matching." *Journal of Machine Learning Research* 6.4 (2005).
  - $= \int_{\xi \in R^n} p_x(\xi)\frac{1}{2}\left|\left|\psi(\xi;\theta)\right|\right|^2 d\xi + \sum_i \int \frac{\partial \psi_i(\xi;\theta)}{\partial \xi_i} p_x(\xi)d\xi + C = \int_{\xi \in R^n} p_x(\xi)\sum_i[\partial\psi_i(\xi;\theta) + \frac{1}{2}\psi_i(\xi;\theta)^2]\, d\xi + C$
- Monte-Carlo Sampling version of $J(\theta)$
  - $\tilde{J}(\theta) = \frac{1}{T}\sum_t \sum_i[\partial\psi_i(x_t;\theta) + \frac{1}{2}\psi_i(x_t;\theta)^2]$
- In its current form, this is designed to be an inference algorithm for known distributions → what if implicit distribution?

Score Matching for Implicit Distribution

- $J(\theta) = \int_{\xi \in R^n} p_x(\xi) \sum_i [\partial \psi_i(\xi;\theta) + \frac{1}{2}\psi_i(\xi;\theta)^2]\, d\xi + C = E_{p_x}\left[tr\left(\nabla_x s(x;\theta)\right) + \frac{1}{2}\left|\left|s(x;\theta)\right|\right|^2\right] + C$
  - Let's say ψ and s are the score functions
  - $\tilde{J}(\theta) = \frac{1}{T}\sum_t \sum_i [\partial \psi_i(x_t;\theta) + \frac{1}{2}\psi_i(x_t;\theta)^2] = \frac{1}{N}\sum_{i=1}^{n}[tr\left(\nabla_x s(x_i;\theta)\right) + \frac{1}{2}\left|\left|s(x_i;\theta)\right|\right|^2]$
  - $\nabla_x s(x_i;\theta) = \nabla_x^2 \log \tilde{p}(x;\theta)$ becomes the Hessian matrix of the modeled density function
  - This causes a problem of the computational complexity
    - $tr\left(\nabla_x s(x;\theta)\right)$ is computing the trace value of the Hessian matrix
    - Therefore, we need another trick to reduce the computational complexity
      - Denoising score matching
      - Sliced score matching
        - $E_{p_v} E_{p_d}\left[v^t \nabla_x s_\theta(x) v + \frac{1}{2}\left|\left|s_\theta(x)\right|\right|^2\right]$
        - p_v : simple known distribution with limited dimensions
        - v : becomes the projection of the Hessian matrix
- From the reparameterization perspective, the score matching becomes
  - $\nabla_\theta H(q_\theta) = -\nabla_\theta E_{q_\theta(x)}[\log q_\theta(x)] = -\nabla_\theta E_{p(\epsilon)}\left[\log q_\theta\left(g_\theta(\epsilon)\right)\right] = -E_{p(\epsilon)}\left[\nabla_x \log q_\theta\left(g_\theta(\epsilon)\right) \nabla_\theta g_\theta(\epsilon)\right]$
    - $q_\theta(x)$: target density to model
    - $g_\theta(\epsilon)$: reparametrization function with error ϵ from trivial distribution

Connection between Score Matching and Denoising Autoencoder (1)

- $J(\theta) = \frac{1}{2}\int_{\xi \in R^n} p_x(\xi) \left|\left|\psi(\xi;\theta) - \psi_x(\xi)\right|\right|^2 d\xi$
  - Explicit format of the density in q : $\psi(\xi;\theta) = \nabla_\xi \log p(\xi;\theta) = \nabla_\xi \log q(\xi;\theta)$
    - $J(\theta) = \int_{\xi \in R^n} p_x(\xi) \sum_i [\partial\psi_i(\xi;\theta) + \frac{1}{2}\psi_i(\xi;\theta)^2]\, d\xi$
    - $\tilde{J}(\theta) = \frac{1}{T}\sum_t \sum_i [\partial\psi_i(x_t;\theta) + \frac{1}{2}\psi_i(x_t;\theta)^2]$
- How to setup q is not determined, yet
  - Let's set $q_\sigma(\tilde{x}, x) = q_\sigma(\tilde{x}|x) q_0(x)$
    - This becomes a kernel of denoising x with the parameterized noise of σ
  - Subsequently,
  - $J_{DSM}(\theta) = E_{q_\sigma(x,\tilde{x})}\left[\frac{1}{2}\left|\psi(\tilde{x};\theta) - \frac{\partial \log q_\sigma(\tilde{x}|x)}{\partial \tilde{x}}\right|^2\right]$
    - $\frac{\partial \log q_\sigma(\tilde{x}|x)}{\partial \tilde{x}} = \frac{1}{\sigma^2}(x - \tilde{x})$
  - Then, the question becomes how to model $\psi(\tilde{x};\theta)$
    - Which should be learnable and flexible enough to learn the perturbed data score

Connection between Score Matching and Denoising Autoencoder (2)

- $J_{DSM}(\theta) = E_{q_\sigma(x,\tilde{x})}\left[\frac{1}{2}\left|\psi(\tilde{x};\theta) - \frac{\partial \log q_\sigma(\tilde{x}|x)}{\partial \tilde{x}}\right|^2\right]$
  - $\frac{\partial \log q_\sigma(\tilde{x}|x)}{\partial \tilde{x}} = \frac{1}{\sigma^2}(x - \tilde{x})$
- $\psi_i(x;\theta) = \frac{\partial \log p(x;\theta)}{\partial x_i} = \frac{1}{\sigma^2}(W^T sigmoid(Wx) - x)$
  - A single-layered denoising autoencoder structure without any constant terms or intercepts
- $J_{DSM}(\theta) = E_{q_\sigma(x,\tilde{x})}\left[\frac{1}{2}\left|\frac{1}{\sigma^2}(W^T sigmoid(W\tilde{x}) - \tilde{x}) - \frac{1}{\sigma^2}(x - \tilde{x})\right|^2\right]$

$$= \frac{1}{2}\frac{1}{\sigma^4}E_{q_\sigma(x,\tilde{x})}[|(W^T sigmoid(W\tilde{x}) - \tilde{x}) - (x - \tilde{x})|^2]$$

$$= \frac{1}{2}\frac{1}{\sigma^4}E_{q_\sigma(x,\tilde{x})}[|W^T sigmoid(W\tilde{x}) - x|^2] = \frac{1}{2}\frac{1}{\sigma^4}J_{DAE}(\theta)$$

  - J_{DSM} becomes the loss function of the denoising autoencoder
- This shows that we can utilize the denoising score matching
  - As a neural network learning function in the generative model
  - Also, the independence between the noise dimension removes the trace of the Hessian matrix

Langevin Dynamics

- $\frac{\partial \log q_\sigma(\tilde{x}|x)}{\partial \tilde{x}} = \frac{1}{\sigma^2}(x - \tilde{x})$
  - This means the transition between x and $\tilde{x}$
  - Ideally, σ can be very small, so learned $\psi_i(x;\theta)$ can well approximate the denoising autoencoder
    - However, this means that we cannot assume $\tilde{x}$ to be the latent space, where we can freely sample
- Therefore, the score-matching needs to be chained
  - How to generate a data instance from a continuous chain
  - Moreover, the current chaining requires a stochastic element of the perturbation

  → Continuous simulation with perturbation
- Langevin method
  - $\tilde{x}_t = \tilde{x}_{t-1} + \frac{\epsilon}{2}\nabla_x \log p(\tilde{x}_{t-1}) + \sqrt{\epsilon} z_t$
    - $z_t \sim N(0, I)$
    - $\psi_i(x;\theta) = \frac{\partial \log p(x;\theta)}{\partial x_i}$
  - Which can be learned from the denoising error
- Finally, the question becomes how to chain $q_\sigma(\tilde{x}|x)$: particularly, how to setup σ

Noise Conditional Score Network

- How to setup σ
  - Let's set $\{\sigma_i\}_{i=1}^{L}$ to satisfy $\frac{\sigma_1}{\sigma_2} = \cdots = \frac{\sigma_{L-1}}{\sigma_L} > 1$
- How to setup q_σ
  - Following the previous setup, $\frac{\partial \log q_\sigma(\tilde{x}|x)}{\partial \tilde{x}} = \frac{1}{\sigma^2}(x - \tilde{x}) = -\frac{1}{\sigma^2}(\tilde{x} - x)$
  - $q_\sigma(x) = \int p_d(t) N(x|t, \sigma^2 I) dt$
- Loss structure of NCSN
  - $L_l(\theta; \sigma_l) = \frac{1}{2} E_{p_d} E_{\tilde{x} \sim N(x, \sigma_l^2 I)} \left[\left\| s_\theta(\tilde{x}, \sigma_l) + \frac{\tilde{x} - x}{\sigma_l^2} \right\|^2 \right]$: This is designed to be a single step
  - $L(\theta; \{\sigma_i\}_{i=1}^{L}) = \frac{1}{L} \sum_{l=1}^{L} \lambda(\sigma_l) \frac{1}{2} E_{p_d} E_{\tilde{x} \sim N(x, \sigma_l^2 I)} \left[\left\| s_\theta(\tilde{x}, \sigma_l) + \frac{\tilde{x} - x}{\sigma_l^2} \right\|^2 \right]$
    - $\lambda(\sigma_l)$: coefficient function, i.e. $\lambda(\sigma) = \sigma^2$
      - Under this choice : $L(\theta; \{\sigma_i\}_{i=1}^{L}) = \frac{1}{L} \sum_{l=1}^{L} \frac{1}{2} E_{p_d} E_{\tilde{x} \sim N(x, \sigma_l^2 I)} \left[\left\| \sigma s_\theta(\tilde{x}, \sigma_l) + \frac{\tilde{x} - x}{\sigma_l^2} \right\|^2 \right]$
        - $\frac{\tilde{x} - x}{\sigma_l} \sim N(0, I)$
        - $s_\theta(\tilde{x}, \sigma_l) \propto \frac{1}{\sigma}$, if s_θ is well trained

Sample Generations from NCSN

- Langevin dynamics is used
  - $\tilde{x}_t = \tilde{x}_{t-1} + \frac{\epsilon}{2}\nabla_x \log p(\tilde{x}_{t-1}) + \sqrt{\epsilon} z_t$
    - $z_t \sim N(0, I)$
    - $\psi_i(x;\theta) = \frac{\partial \log p(x;\theta)}{\partial x_i}$
  - Given the denoising function : $\psi_i(x;\theta) = s_\theta(x,\sigma)$
- α_i : the Langevin dynamics step magnitude
  - $\{\sigma_i\}_{i=1}^{L}$ to satisfy $\frac{\sigma_1}{\sigma_2} = \cdots = \frac{\sigma_{L-1}}{\sigma_L} > 1$
    - $\sigma_1 > \sigma_2 \ldots > \sigma_L$
    - Given the current convention
      - $\tilde{x}_0$ is the latent sample of noised
      - $\tilde{x}_T$ is the denoised sample
    - In most noised samples (pure latent of $N(0, I)$), σ will be high
    - Near the data distribution, σ will be low
  - $\alpha_i = \epsilon \frac{\sigma_i^2}{\sigma_L^2}$
    - In most noise samples, the α_i step will be large

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**
  $\alpha_i \leftarrow \epsilon \cdot \sigma_i^2 / \sigma_L^2$ $\triangleright$ 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 $\tilde{\mathbf{x}}_T$

Comparison between Diffusion and NCSN

- Loss structure
  - DDPM : $L(\theta) = \frac{1}{T}\sum_{1\le t<T} p(t) E_{x_0\sim p(x_0), \epsilon\sim N(0,I)}\left[\left|\left|\epsilon - \epsilon_\theta(x_t, t)\right|\right|^2\right]$
  - NCSN : $L\left(\theta; \{\sigma_i\}_{i=1}^L\right) = \frac{1}{L}\sum_{l=1}^L \lambda(\sigma_l)\frac{1}{2}E_{p_d}E_{\tilde{x}\sim N(x,\sigma_l^2 I)}\left[\left|\left|s_\theta(\tilde{x},\sigma_l) + \frac{\tilde{x}-x}{\sigma_l^2}\right|\right|^2\right]$
- Generation algorithm
  - $\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{(1-\bar{\alpha}_t)}}\epsilon_\theta(x_t, t)\right)$
  - $\tilde{x}_t = \tilde{x}_{t-1} + \frac{\epsilon}{2}\nabla_x \log p(\tilde{x}_{t-1}) + \sqrt{\epsilon} z_t$

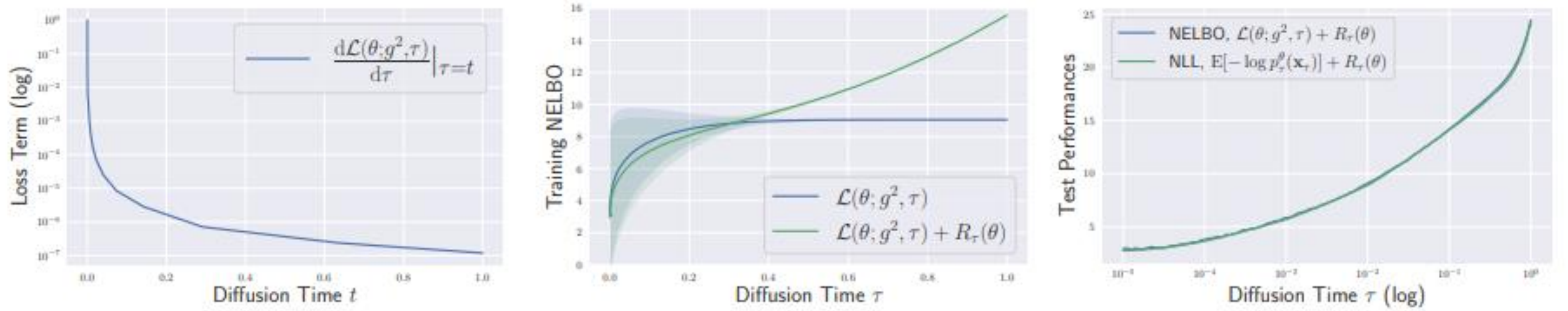
Algorithm 2 Sampling

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

Algorithm 1 Annealed Langevin dynamics.

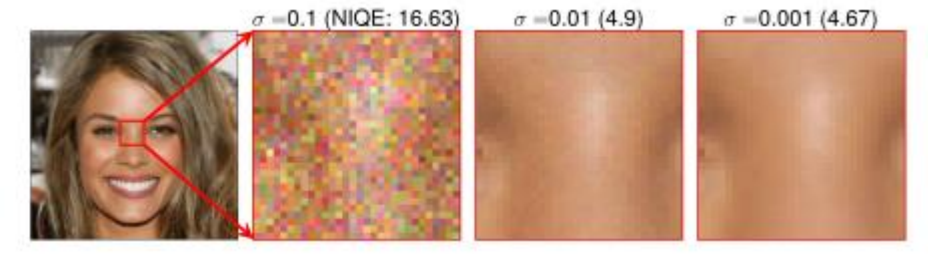
Require: $\{\sigma_i\}_{i=1}^L, \epsilon, T$.
1: Initialize $\tilde{\mathbf{x}}_0$
2: **for** $i \leftarrow 1$ to L **do**
3: $\quad \alpha_i \leftarrow \epsilon \cdot \sigma_i^2/\sigma_L^2$ $\quad \triangleright$ α_i is the step size.
4: $\quad$ **for** $t \leftarrow 1$ to T **do**
5: $\quad\quad$ Draw $\mathbf{z}_t \sim \mathcal{N}(0, I)$
6: $\quad\quad \tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \frac{\alpha_i}{2}\mathbf{s}_\theta(\tilde{\mathbf{x}}_{t-1}, \sigma_i) + \sqrt{\alpha_i}\,\mathbf{z}_t$
7: $\quad$ **end for**
8: $\quad \tilde{\mathbf{x}}_0 \leftarrow \tilde{\mathbf{x}}_T$
9: **end for**
return $\tilde{\mathbf{x}}_T$

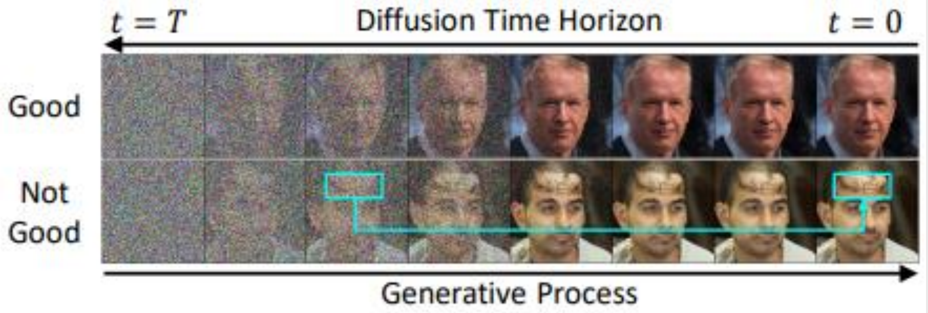
Criticality of Noise Scheduling



(a) Integrand by Time (b) Variational Bound Truncated at τ (c) Test Performance by Log-Time

- Small diffusion time dominates the loss term
  - $\frac{\partial \log q_\sigma(\tilde{x}|x)}{\partial \tilde{x}} = \frac{1}{\sigma^2}(x-\tilde{x})$, Very small σ
  - The matching between scores become large magnitude
- What happens when σ is either very small/large?
- Possible remedy?
  - Magnitude vs. # of samplings



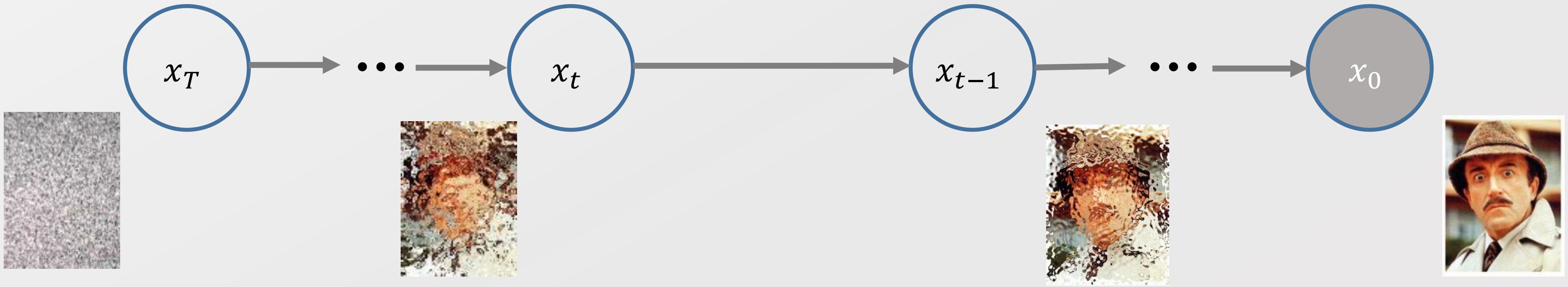


$$\mathcal{L}(\boldsymbol{\theta};g^2,\epsilon)$$
$$= \frac{Z_\epsilon}{2}\int_\epsilon^T p_{iw}(t)\sigma^2(t)\mathbb{E}\left[\|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)-\nabla\log p_{0t}(\mathbf{x}_t|\mathbf{x}_0)\|_2^2\right]\mathrm{d}t$$
$$\approx \frac{Z_\epsilon}{2B}\sum_{b=1}^{B}\sigma^2(t_{iw}^{(b)})\left\|\mathbf{s}_{\boldsymbol{\theta}}\left(\mathbf{x}_{t_{iw}^{(b)}},t_{iw}^{(b)}\right)-\frac{\boldsymbol{\epsilon}^{(b)}}{\sigma(t_{iw}^{(b)})}\right\|_2^2$$

where $\{t_{iw}^{(b)}\}_{b=1}^{B}$ is the Monte-Carlo sample from the importance distribution, i.e., $t_{iw}^{(b)} \sim p_{iw}(t) \propto \frac{g^2(t)}{\sigma^2(t)}$.

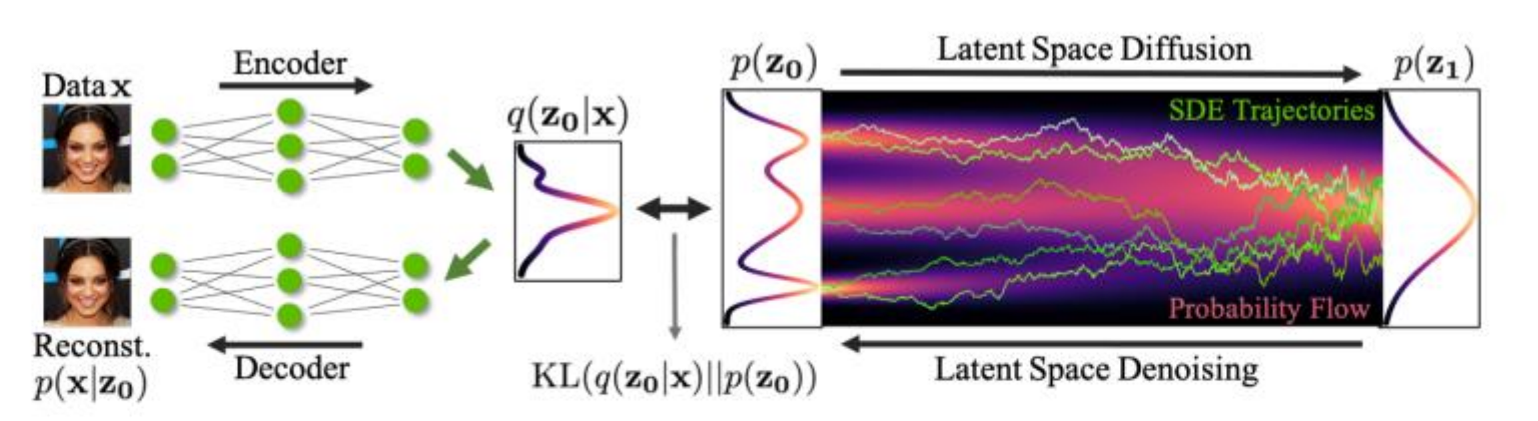
Linear Encoding of Diffusion Models

- Linear encodings of DDPM and NCSN
  - NCSN : $\frac{\partial \log q_\sigma(\tilde{x}|x)}{\partial \tilde{x}} = \frac{1}{\sigma^2}(x - \tilde{x})$
  - DDPM : $q(x_t|x_{t-1}) = N\left(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I\right)$
- Why linear encoding?
  - NCSN : the trace of the Hessian matrix
  - DDPM : the closed-form solution of the perturbed features
- How to make the linear encoding to be non-linear
  - Make the feature to be transformed to the embedding through nonlinear encoding



VAE + Diffusion Model (1)

- Very simple idea
  - Add a VAE structure
  - Define diffusion model upon z, not x
- Natural expansion, given the ELBO structure of diffusion models



$$\mathcal{L}(\mathbf{x},\phi,\theta,\psi) = \mathbb{E}_{q_\phi(\mathbf{z}_0|\mathbf{x})}\left[-\log p_\psi(\mathbf{x}|\mathbf{z}_0)\right] + \mathrm{KL}\left(q_\phi(\mathbf{z}_0|\mathbf{x})||p_\theta(\mathbf{z}_0)\right)$$

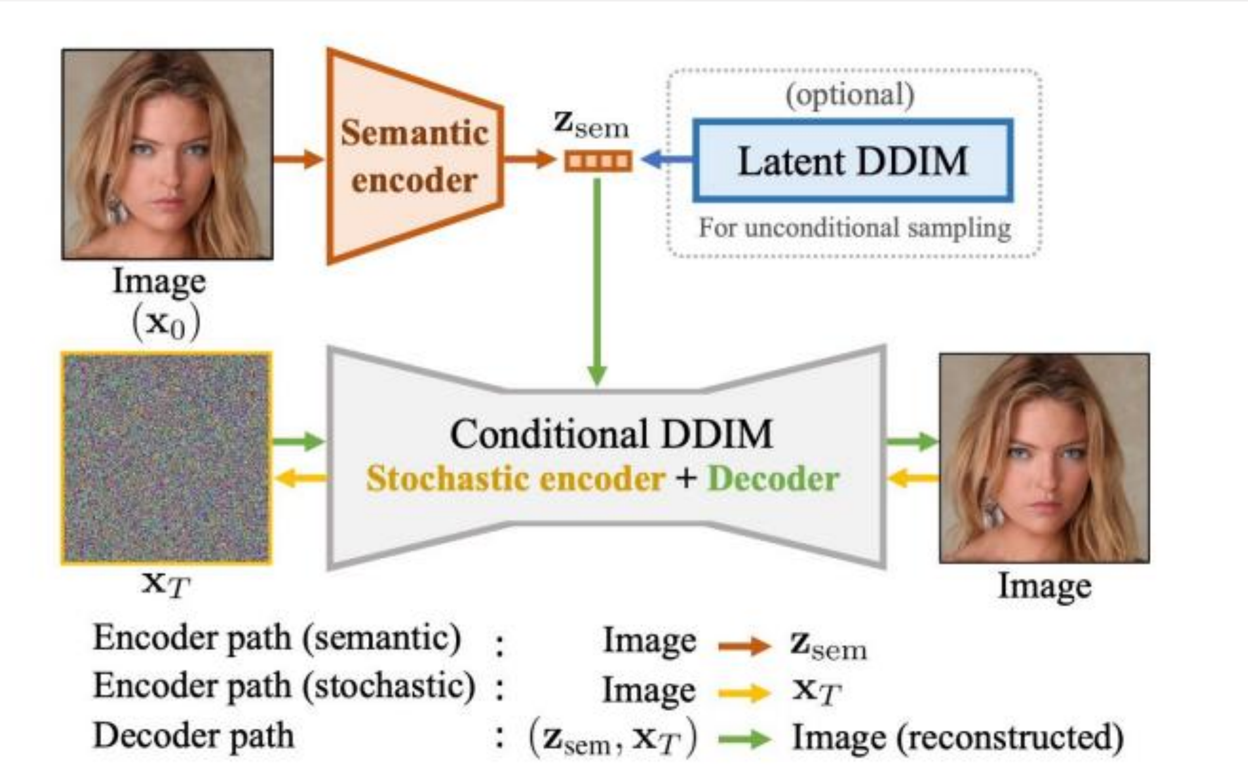
$$= \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}_0|\mathbf{x})}\left[-\log p_\psi(\mathbf{x}|\mathbf{z}_0)\right]}_{\text{reconstruction term}} + \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}_0|\mathbf{x})}\left[\log q_\phi(\mathbf{z}_0|\mathbf{x})\right]}_{\text{negative encoder entropy}} + \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}_0|\mathbf{x})}\left[-\log p_\theta(\mathbf{z}_0)\right]}_{\text{cross entropy}}$$

Theorem 1. *Given two distributions $q(\mathbf{z}_0|\mathbf{x})$ and $p(\mathbf{z}_0)$, defined in the continuous space $\mathbb{R}^D$, denote the marginal distributions of diffused samples under the SDE in Eq. 1 at time t with $q(\mathbf{z}_t|\mathbf{x})$ and $p(\mathbf{z}_t)$. Assuming mild smoothness conditions on $\log q(\mathbf{z}_t|\mathbf{x})$ and $\log p(\mathbf{z}_t)$, the cross entropy is:*

$$CE(q(\mathbf{z}_0|\mathbf{x})||p(\mathbf{z}_0)) = \mathbb{E}_{t\sim\mathcal{U}[0,1]}\left[\frac{g(t)^2}{2}\mathbb{E}_{q(\mathbf{z}_t,\mathbf{z}_0|\mathbf{x})}\left[||\nabla_{\mathbf{z}_t}\log q(\mathbf{z}_t|\mathbf{z}_0) - \nabla_{\mathbf{z}_t}\log p(\mathbf{z}_t)||_2^2\right]\right] + \frac{D}{2}\log\left(2\pi e\sigma_0^2\right).$$

VAE + Diffusion Model (2)

- DDPM
  - $L(\theta) = \frac{1}{T} \sum_{1 \le t < T} p(t) E_{x_0 \sim p(x_0), \epsilon \sim N(0,I)} \left[\left| \left| \epsilon - \epsilon_\theta(x_t, t) \right| \right|^2 \right]$
- Utilizing the conditional structure of DDIM
  - Mixing the semantic latent variable z_{sem}
  - Error pattern estimation is updated to anticipate both time and semantics



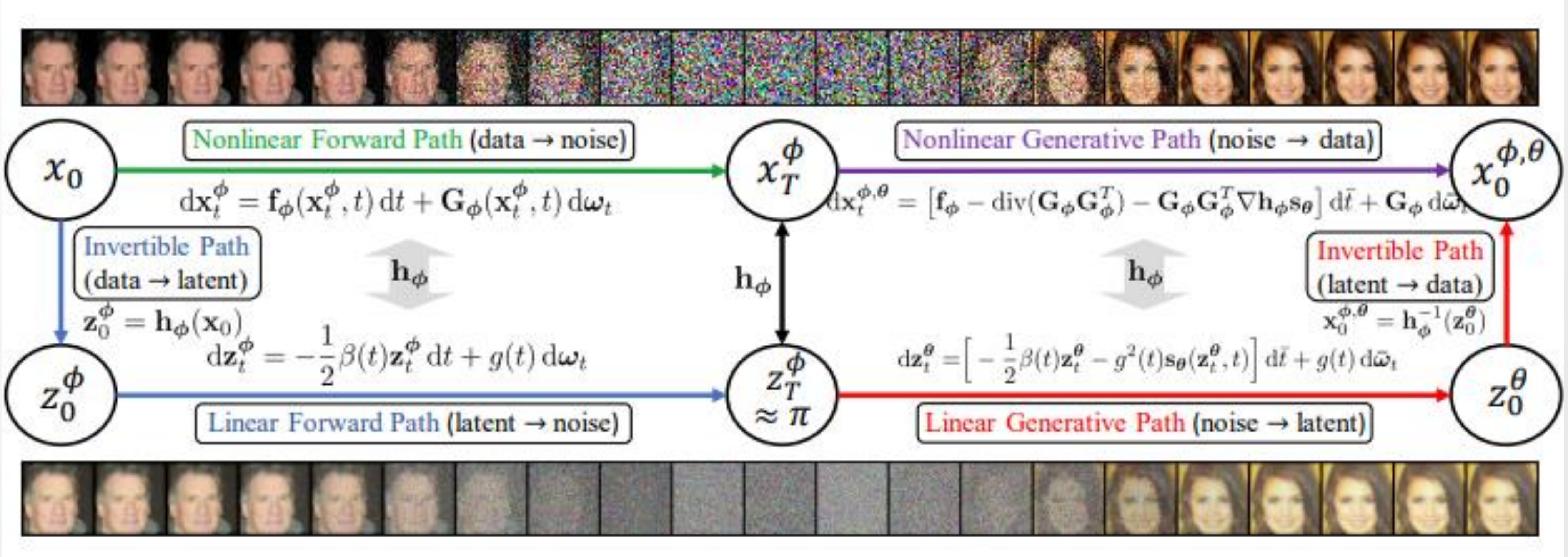
$$p_\theta(\mathbf{x}_{0:T} \mid \mathbf{z}_{\text{sem}}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{z}_{\text{sem}}) \qquad (3)$$

$$p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{z}_{\text{sem}}) = \begin{cases} \mathcal{N}(\mathbf{f}_\theta(\mathbf{x}_1, 1, \mathbf{z}_{\text{sem}}), \mathbf{0}) & \text{if } t = 1 \\ q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{f}_\theta(\mathbf{x}_t, t, \mathbf{z}_{\text{sem}})) & \text{otherwise} \end{cases} \qquad (4)$$

$$\mathbf{x}_{t+1} = \sqrt{\alpha_{t+1}} \mathbf{f}_\theta(\mathbf{x}_t, t, \mathbf{z}_{\text{sem}}) + \sqrt{1 - \alpha_{t+1}} \epsilon_\theta(\mathbf{x}_t, t, \mathbf{z}_{\text{sem}})$$

$$L_{\text{latent}} = \sum_{t=1}^{T} \mathbb{E}_{\mathbf{z}_{\text{sem}}, \epsilon_t} \left[\| \epsilon_\omega(\mathbf{z}_{\text{sem},t}, t) - \epsilon_t \|_1 \right]$$

Flow + Diffusion Model



- Latent space diffusion
  - $dz_t^\phi = -\frac{1}{2}\beta(t) z_t^\phi dt + g(t) dw_t$
- Forward "data" diffusion
  - $dx_t^\phi = f_\phi\left(x_t^\phi, t\right) dt + G_\phi\left(x_t^\phi, t\right) dw_t$
- Now, we have a diffusion term with a learning function of $G_\phi\left(x_t^\phi, t\right)$