Denoising Diffusion Probabilistic Models

Jun Hyung Lee

Korea University
Department of
Computer Science and Engineering

IE LAB
Interlligence Engineering

References

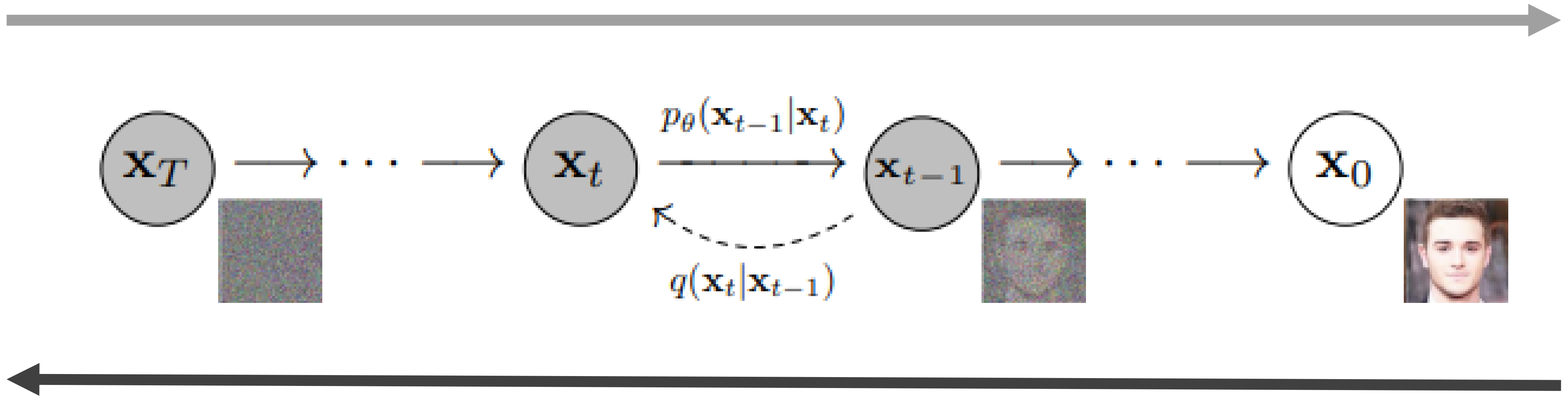
- **Deep Unsupervised Learning using Nonequilibrium Thermodynamics,** Jasacha Sohl-Dickstein (2015)
- **Generative Modeling by Estimating Gradients of the Data Distribution,** Yang Song, Stefano Ermon (2020)
- **https://huggingface.co/blog/annotated-diffusion**
- **https://lilianweng.github.io/posts/2021-07-11-diffusion-models/**
- **Denoising Diffusion-based Generative Modeling: Foundations and Applications,** Karsten Kreis, Ruiqi Gao, Arash Vahdat (CVPR Diffusion Tutorial 2022)
- **https://angusturner.github.io/generative_models/2021/06/29/diffusion-probabilistic-models-I.html**
- **https://www.assemblyai.com/blog/diffusion-models-for-machine-learning-introduction/**

Denoising Diffusion Models

Denoising Diffusion Models contain two processes:

① Forward Diffusion Process: Process that gradually adds noise to input data $\boldsymbol{q(x_t|x_{t-1})}$ -> fixed, no learnable parameters
② Denoising Reverse Process: Process that trains to generate data by denoising $\mathbf{P_\theta(x_{t-1}|x_t)}$ -> training, generating procedure

Denoising Reverse Process (Generative)



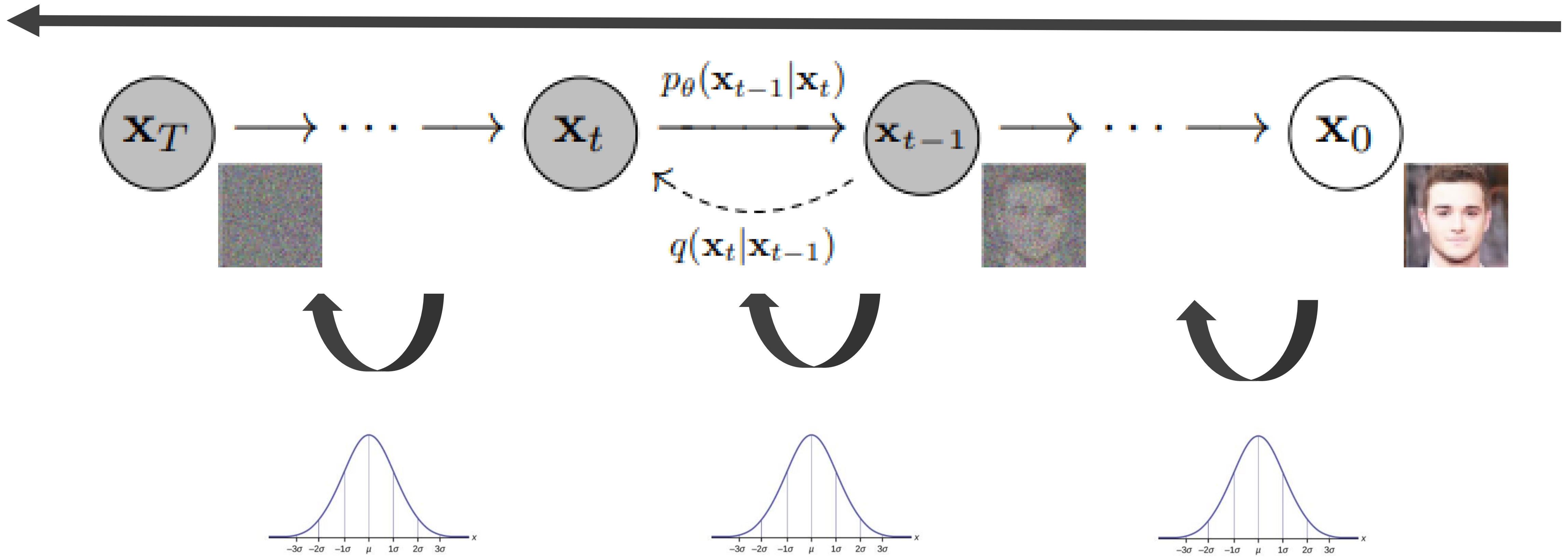
Forward Diffusion Process (Fixed)

Forward Diffusion Process

- Denoising Diffusion Models contain two processes:
  Forward Diffusion Process: Process that gradually adds noise to input data $\boldsymbol{q(x_t|x_{t-1})}$ -> fixed, no learnable parameters
- x_T nearly becomes Isotropic Gaussian

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

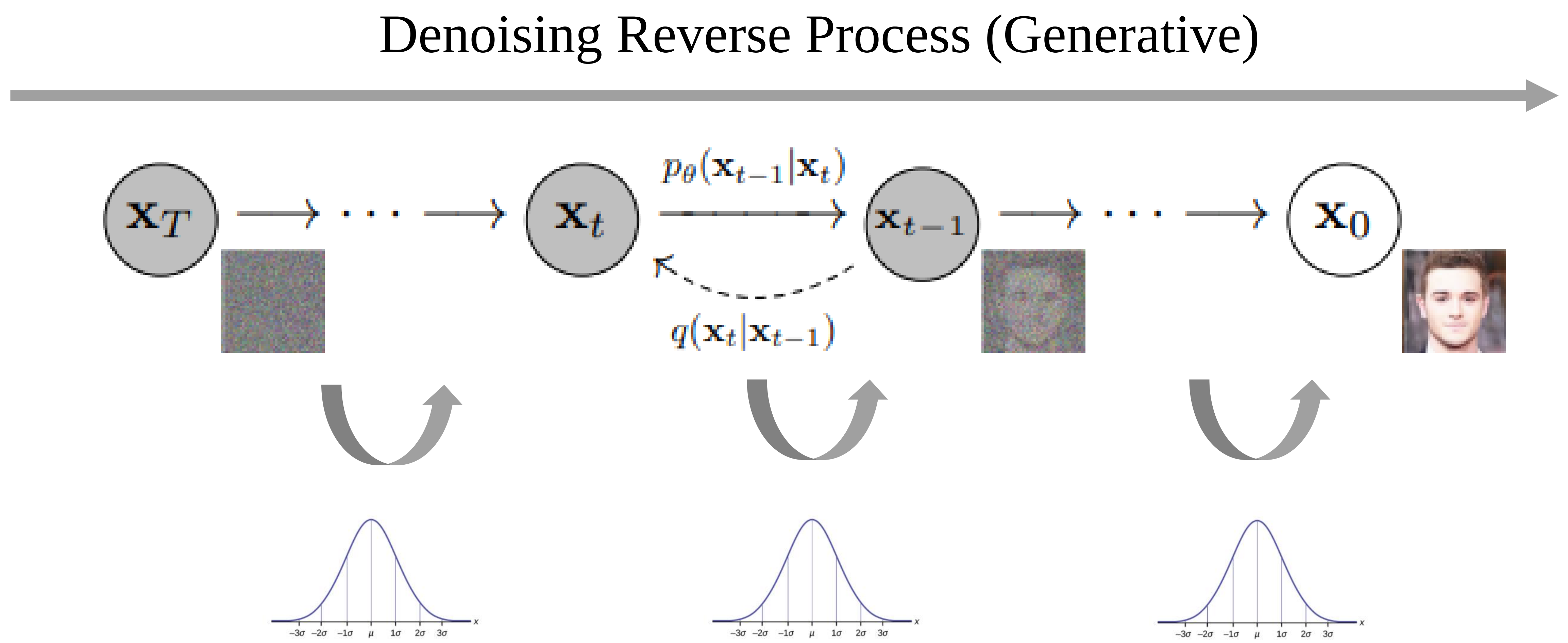
Forward Diffusion Process (Fixed)



Denoising Reverse Process

- If we can reverse the forward process $\boldsymbol{q}(\boldsymbol{x_{t-1}}|\boldsymbol{x_t})$ we will be able to recreate the true sample from a gaussian noise input. However, it's not an easy task to estimate the true sample because it requires to use the entire dataset. Therefore, we need to learn a model P_θ to approximate these conditional probabilities.

$$P_\theta(x_{0:T}) \coloneqq P(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t) \longrightarrow P_\theta(x_{t-1}|x_t) \coloneqq N\big(x_{t-1}; \mu_\theta(x_t,t), \Sigma_\theta(x_t,t)\big) q(x_{1:T}|x_0) \coloneqq \prod_{t=1}^{T} q(x_t|x_{t-1})$$



Forward Process sampling at x_t

- The Markov Chain of two diffusion processes can sample x_t at any arbitrary time step t in a closed form using reparameterization trick.

Let $\boldsymbol{\alpha_t = 1 - \beta_t}$ and $\boldsymbol{\bar{\alpha}_t = \prod_{i=1}^{T} \alpha_i}$:

$$x_t = \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}z_{t-1} \quad \text{; where } z_{t-1}, z_{t-2} \ldots N(0, I)$$

$$x_t = \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\bar{z}_{t-2} \quad \text{; where } \bar{z}_{t-2} \text{ merges two Gaussians}$$

$$\vdots$$

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}z$$

$$\underline{q(x_t|x_0) = N\left(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I\right)}$$

https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Variational Bound

Loss Terms:

① L_T has no learnable parameters (constant during training)
② L_{t-1} are KL divergence between gaussians (KL divergence between Forward Process Posterior and Denoising Reverse Process)
③ L_0 is the familiar reconstruction term

$$L := E_q\left[\underbrace{D_{kL}\big(q(x_T|x_0)||p(x_T)\big)}_{L_T} + \sum_{t>1} \underbrace{D_{kL}(q(x_{t-1}|x_t, x_0)||P_\theta(x_{t-1}|x_t))}_{L_{t-1}} \underbrace{- \log P_\theta(x_0|x_1)}_{L_0}\right]$$

https://angusturner.github.io/generative_models/2021/06/29/diffusion-probabilistic-models-I.html

Mean Predictor (L_{t-1})

- Trainable network in reverse denoising process:

$$P_\theta(x_{t-1}|x_t) = N(x_{t-1}; \underbrace{\boldsymbol{\mu_\theta}(\boldsymbol{x_t}, \mathbf{t})}_{\text{Trainable network}}, \sigma_t^2 I)$$

- Where $q(x_{t-1}|x_t, x_0)$, it becomes tractable when conditioned on x_0 :

$$q(x_{t-1}|x_t, x_0) = N\left(x_{t-1}; \boldsymbol{\tilde{\mu}_t}(\boldsymbol{x_t}, \boldsymbol{x_0}), \tilde{\beta}_t I\right)$$

KL Divergence

- Since both $q(x_{t-1}|x_t, x_0)$ and $P_\theta(x_{t-1}|x_t)$ are gaussians, the KL divergence has a simple form:

$$L_{t-1} = E_q\left[\underbrace{\frac{1}{2\sigma_t^2}||\boldsymbol{\tilde{\mu}_t}(\boldsymbol{x_t}, \boldsymbol{x_0}) - \boldsymbol{\mu_\theta}(\boldsymbol{x_t}, \boldsymbol{t})||^2}_{\textbf{Mean Squared Error}}\right] + \mathrm{C}$$

Forward Process Posterior Mean($\tilde{\mu}_t$)

- So how do we know $\tilde{\boldsymbol{\mu}}_t$?
- Following the standard Gaussian density function, the mean and variance can be parameterized.
  "Reverse conditional probability" is tractable when conditioned on x_0 :

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

- We can represent $\mathbf{x_0} = \frac{\mathbf{1}}{\sqrt{\bar{\mathbf{a}}_\mathbf{t}}}\left(\boldsymbol{x_t} - \sqrt{\mathbf{1} - \bar{\boldsymbol{\alpha}}_\mathbf{t}}\boldsymbol{\varepsilon}\right)$ and plug it into the above equation:

$$\tilde{\boldsymbol{\mu}}_t(x_t, x_0) := \frac{\sqrt{\bar{\alpha}_t - 1}\beta_t}{1-\bar{\alpha}_t} \underbrace{\frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \sqrt{1-\bar{\alpha}_t}\varepsilon\right)}_{x_0} + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} x_t \longrightarrow \tilde{\boldsymbol{\mu}}_t = \frac{\mathbf{1}}{\sqrt{\boldsymbol{\alpha}_\mathbf{t}}}\left(\boldsymbol{x_t} - \frac{\boldsymbol{\beta}_\mathbf{t}}{\sqrt{\mathbf{1} - \bar{\boldsymbol{\alpha}}_\mathbf{t}}}\boldsymbol{\varepsilon}\right)$$

https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Denoising Reverse Process Mean(μ_θ)

- We want to train $\boldsymbol{\mu_\theta}$ to predict $\boldsymbol{\tilde{\mu}_t} = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x_t} - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\varepsilon}\right)$
- x_t is available as input at training time, we can reparameterize the gaussian noise term instead and predict ε_t from the input x_t at time step t :

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

$$\text{Thus,}\ \ x_{t-1} = N\left(x_{t-1}; \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x_t} - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta(x_t, t)\right)\right), \Sigma_\theta(x_t, t)$$

- The loss term L_t is parameterized to minimize the difference from $\boldsymbol{\tilde{\mu}_t}$:

$$L_t = E_{x_0,\varepsilon}\left[\frac{1}{2||\Sigma_\theta(x_t,t)||_2^2}||\tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t,t)||^2\right]$$

$$L_t = E_{x_0,\varepsilon}\left[\frac{1}{2||\Sigma_\theta(x_t,t)||_2^2}||\frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_t\right) - \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta(x_t,t)\right)||^2\right]$$

$$L_t = E_{x_0,\varepsilon}\left[\frac{\beta_t^2}{2\alpha_t(1-\bar{\alpha}_t)||\Sigma_\theta(x_t,t)||_2^2}||\varepsilon_t - \varepsilon_\theta(x_t,t)||^2\right]$$

$$L_t = E_{x_0,\varepsilon}\left[\frac{\beta_t^2}{2\alpha_t(1-\bar{\alpha}_t)||\Sigma_\theta(x_t,t)||_2^2}||\varepsilon_t - \varepsilon_\theta(x_t,t)||^2\right]$$

https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Noise Predictor (L_{t-1})

Mean Predictor -> Noise Predictor

ε-prediction both resembles Langevin dynamics and it simplifies the diffusion model's variational bound to an objective that resembles denoising score matching.

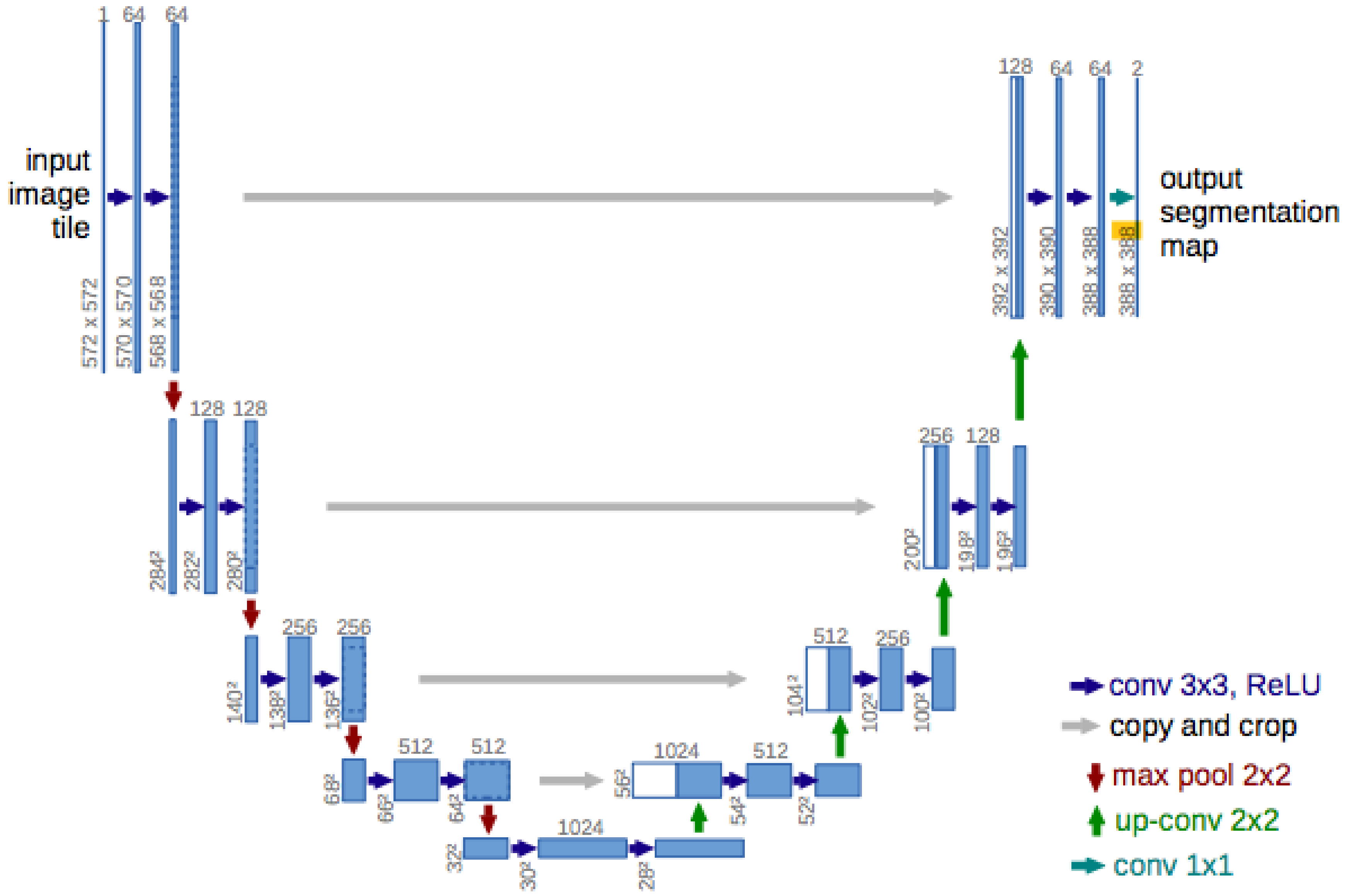
$$E_{x_0,\varepsilon}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}||\varepsilon-\varepsilon_\theta\left(\sqrt{\bar{\alpha}_t}x_0+\sqrt{1-\bar{\alpha}_t}\varepsilon,t\right)||^2\right]$$

$$\Downarrow$$

$$L_{\boldsymbol{simple}}(\theta) := E_{t,x_0,\varepsilon}\left[||\varepsilon-\varepsilon_\theta\left(\sqrt{\bar{\alpha}_t}x_0+\sqrt{1-\bar{\alpha}_t}\varepsilon,t\right)||^2\right]$$

Network Architecture

- Diffusion model has the same input and output dimensions, therefore U-Net-like architectures are commonly implemented.
- Parameters are shared across time, which is specified to the network using the Transformer sinusoidal position embedding and used self-attention at the 16 x 16 feature map resolution.
- Learning reverse process variance leads to unstable training and poorer sample quality compared to fixed variances.
- $\beta_1 = 10^{-4}, \beta_T = 0.02$



Sampling Algorithm

- Algorithm 1 #5 resembles denoising score matching (Song, 2019)
- The connection also has the reverse implication that a certain weighted form of denoising score matching is the same as variational inference to train a Langevin-like sampler.

Algorithm 1 Training

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

Algorithm 2 Sampling

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

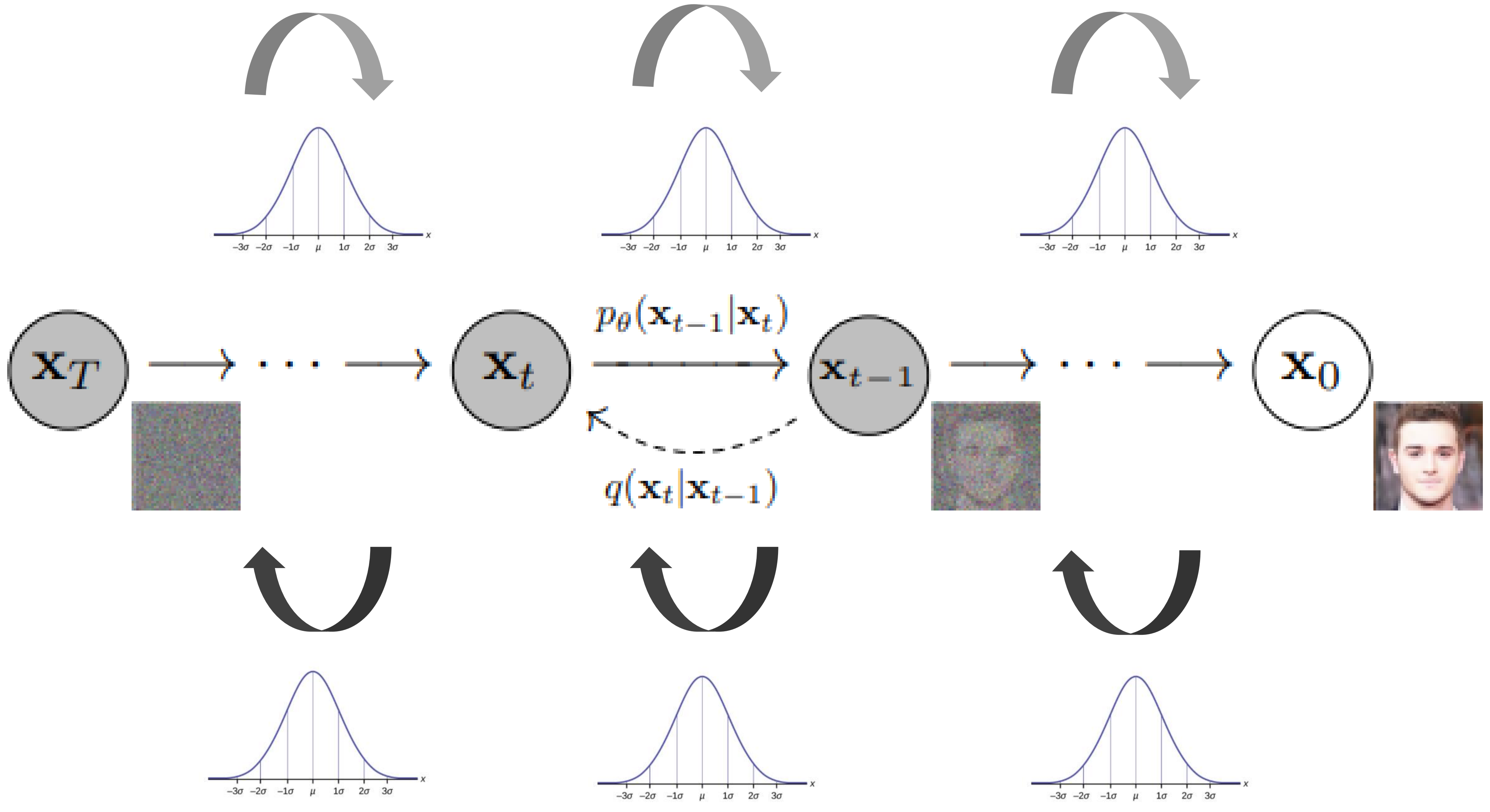
- Other methods for learning transition operators of Markov Chain include infusion training, variational walkback, generative stochastic networks etc

Summary

Denoising Diffusion Models contain two processes:

① Forward Diffusion Process: Process that gradually adds noise to input data $\boldsymbol{q(x_t|x_{t-1})}$ -> fixed, no learnable parameters
② Denoising Reverse Process: Process that trains to generate data by denoising $\mathbf{P_\theta(x_{t-1}|x_t)}$ -> training, generating procedure

Denoising Reverse Process (Generative)



Forward Diffusion Process (Fixed)