Introduction to Deep Generative Modeling

Lecture #15

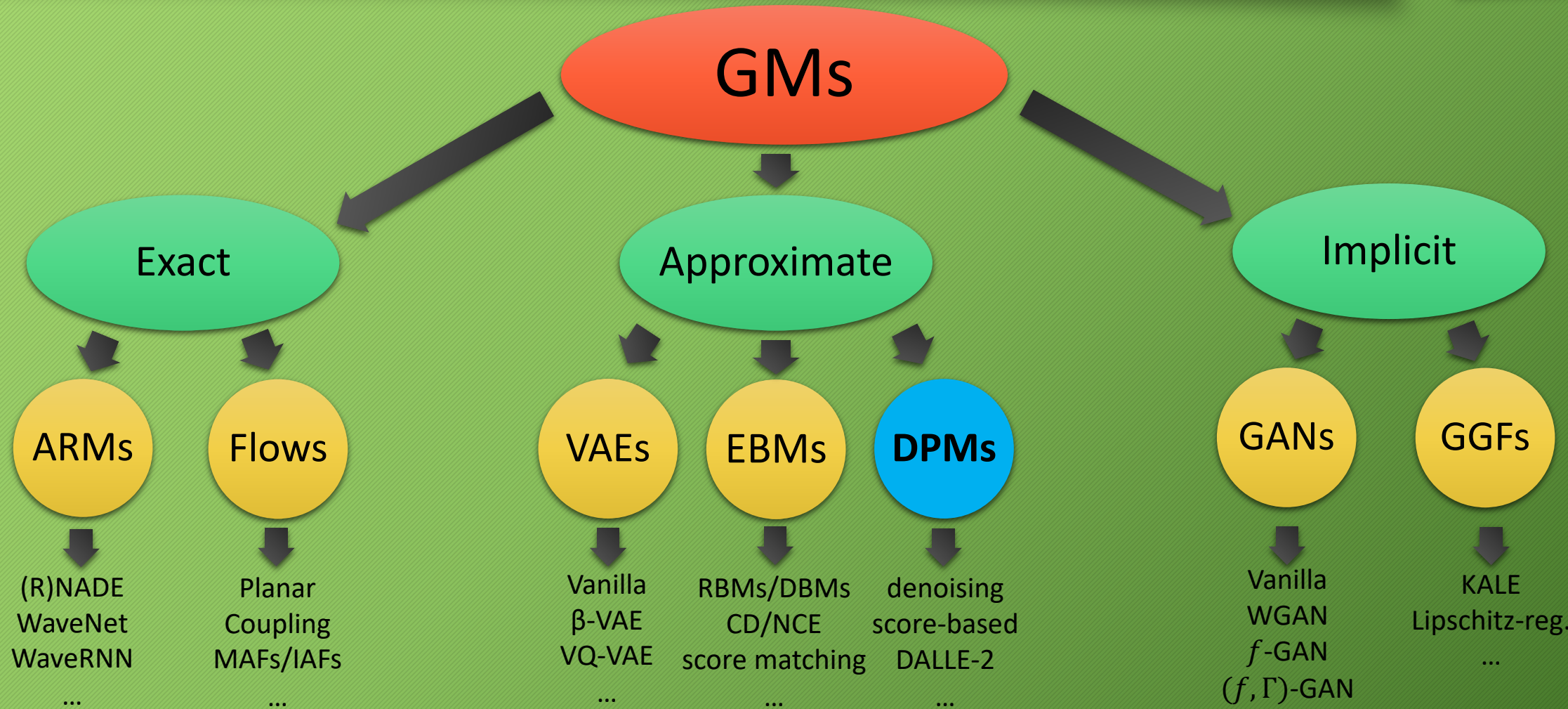
HY-673 – Computer Science Dep, University of Crete

Professors: Yannis Pantazis, Yannis Stylianou

Teaching Assistant: Thomas Marchioro

Taxonomy of GMs

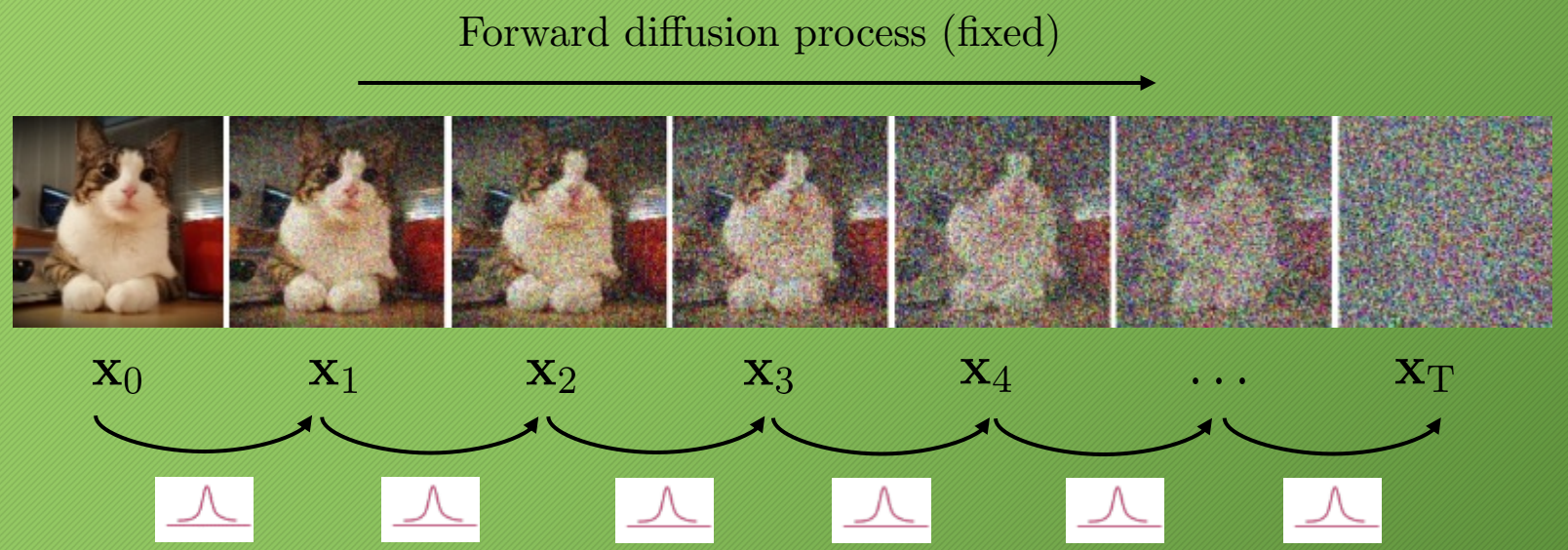
Lecture #15
HY-673



Recap: Forward Diffusion Process

Lecture #15
HY-673

The forward diffusion process:

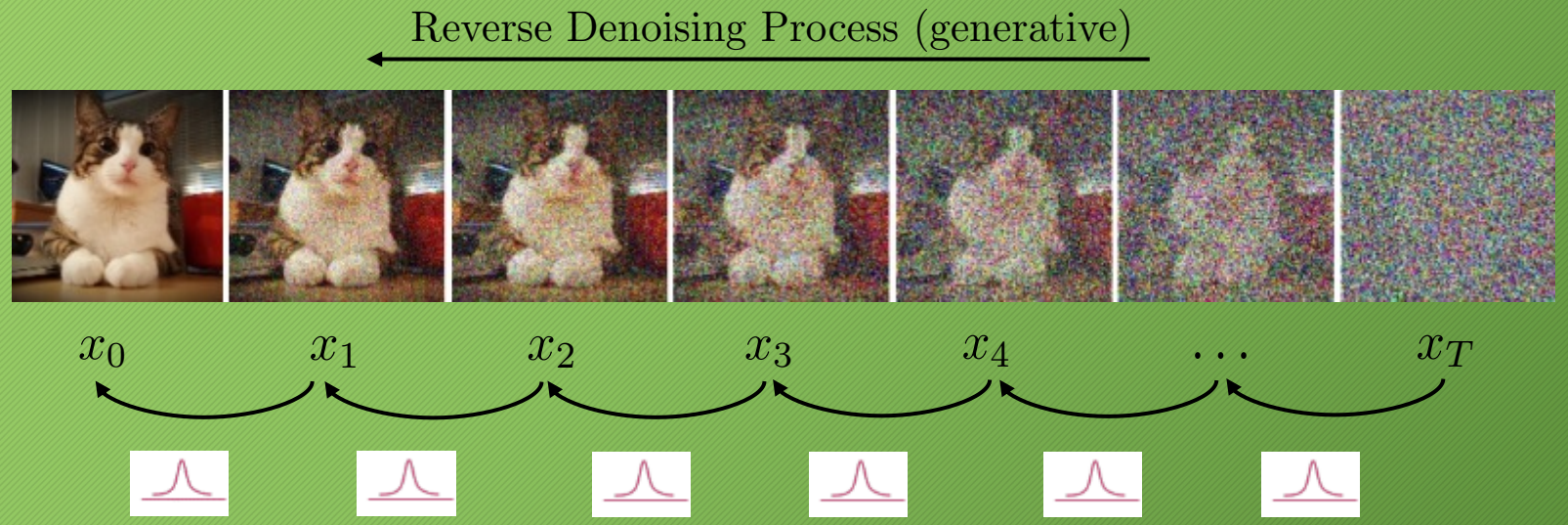


$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

Recap: Reverse Denoising Process

Lecture #15
HY-673

The formal definition of the reverse process in T steps:



$$p(x_T) = \mathcal{N}(x_T; 0, I_d)$$

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \underbrace{\mu_\theta(x_t, t)}_{\text{Trainable network}}, \sigma_t^2 I_d)$$

(U-net, Denoising Autoencoder)

$$\Rightarrow \quad p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t).$$

$$\approx q(x_{t-1}|x_t) \text{ (true posterior; intractable)}$$

Recap: Training and Sampling

Lecture #15
HY-673

Minimize a simplification of negative ELBO:

$$\mathrm{L}_{\text{simple}} = \mathbb{E}_{x_0 \sim p_d(x_0), \epsilon \sim \mathcal{N}(0, I_d), t \sim \mathcal{U}(1, \mathrm{T})} \left[||\epsilon - \epsilon_\theta(\underbrace{\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon}_{x_t}, t)||^2 \right].$$

Algorithm 1 Training

1: **repeat**
2: $x_0 \sim p_d(x_0)$
3: $t \sim \text{Uniform}(1, \ldots, T)$
4: $\epsilon \sim \mathcal{N}(0, I_d)$
5: Take gradient descent step on

$$\nabla_\theta ||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, t)||^2$$

6: **until** converged

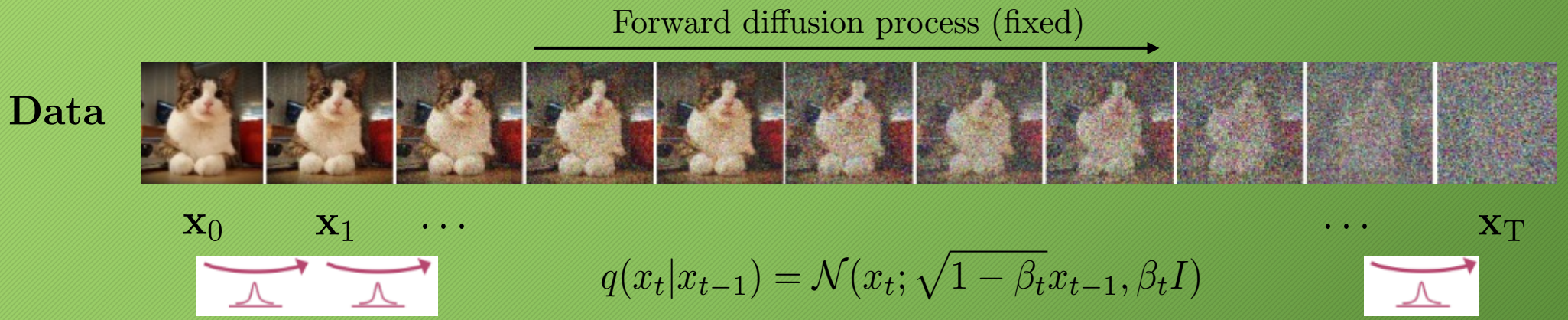
Algorithm 2 Sampling

1: $x_\mathrm{T} \sim \mathcal{N}(0, I_d)$
2: **for** $t = T, \ldots, 1$ **do**
3: $z \sim \mathcal{N}(0, I_d)$
4: $x_{t-1} = \frac{1}{\sqrt{1-\beta_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)) + \sigma_t z$
5: **end for**
6: **return** x_0

Forward Diffusion Process

Lecture #15
HY-673

Consider the limit of **many small steps:**



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

$$x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}z_t. \qquad (z_t \sim \mathcal{N}(0, I))$$

$$= \sqrt{1-\beta(t)\Delta t}x_{t-1} + \sqrt{\beta(t)\Delta t}z_t \qquad (\beta_t := \beta(t)\Delta t)$$

$$\approx x_{t-1} - \frac{\beta(t)\Delta(t)}{2}x_{t-1} + \sqrt{\beta(t)\Delta t}z_t. \qquad \text{(Taylor expansion)}$$

https://arxiv.org/abs/2011.13456

Forward Diffusion Process as Stochastic Differential Equation

Lecture #15
HY-673

Consider the limit of **many small steps:**

Forward diffusion process (fixed)

Data  **Noise**

$\mathbf{x}_0$ $\mathbf{x}_1$ $\cdots$ $\cdots$ $\mathbf{x}_T$

$$x_t - x_{t-1} \approx \frac{\beta(t)\Delta(t)}{2} x_{t-1} + \sqrt{\beta(t)\Delta t} z_t$$

$$dx_t = -\frac{1}{2}\beta(t) x_t dt + \sqrt{\beta(t)} dW_t$$

W_t: Weiner process

Stochastic Differential Equation (SDE)

describing the diffusion process in the infinitesimal limit

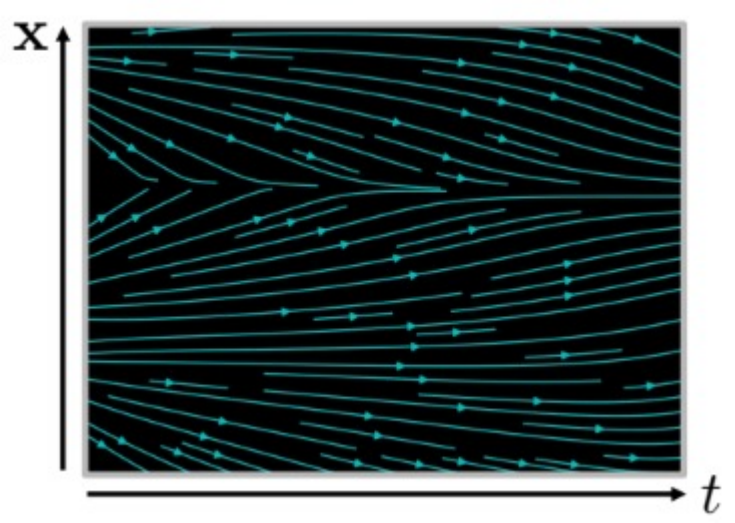
https://arxiv.org/abs/2011.13456

Crash Course in Differential Equations

Lecture #15
HY-673

Ordinary Differential Equation (ODE):

$$\frac{dx}{dt} = f(x,t) \text{ or } dx = f(x,t)dt$$

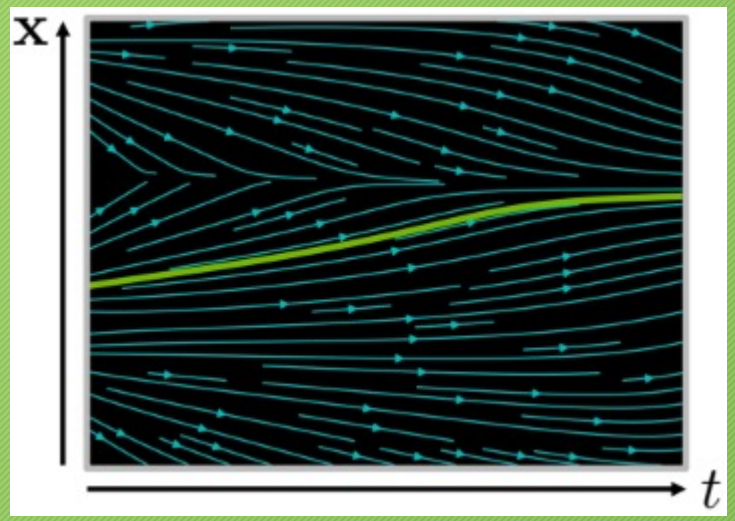


Crash Course in Differential Equations

Lecture #15
HY-673

Ordinary Differential Equation (ODE):

$$\frac{dx_t}{dt} = f(x_t, t) \text{ or } dx_t = f(x_t, t)dt$$



Analytical Solution:

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

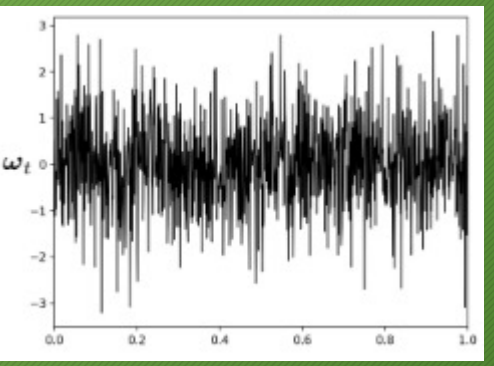
Iterative Numerical Solution:

$$x(t + \Delta t) \approx x(t) + f(x(t), t)\Delta t$$

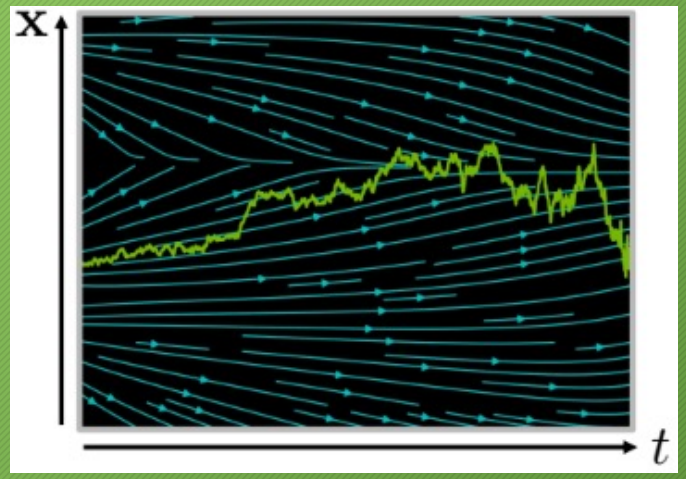
Stochastic Differential Equation (SDE):

$$\frac{dx_t}{dt} = \underbrace{f(x_t, t)}_{\text{drift coefficient}} + \underbrace{\sigma(x_t, t)}_{\text{diffusion coefficient}} W'_t$$

Wiener Process Derivative (i.e., Gaussian White Noise)



$$\left(dx_t = f(x_t, t)dt + \sigma(x_t, t)dW_t \right)$$



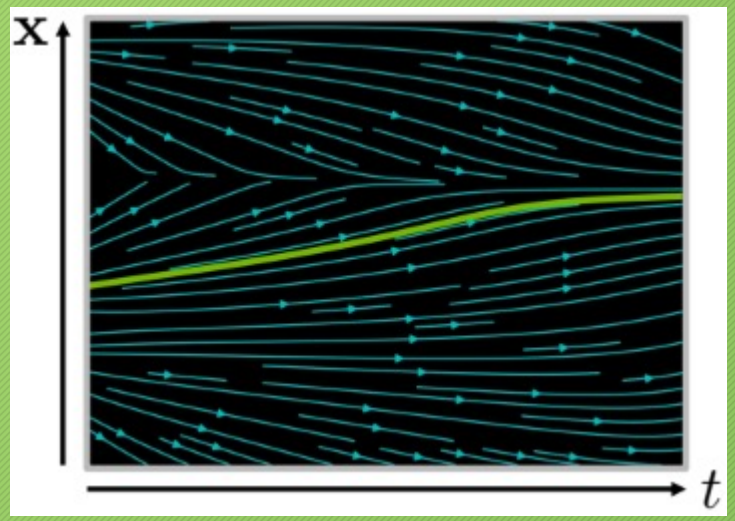
$$x(t + \Delta t) \approx x(t) + f(x(t), t)\Delta t + \sigma(x(t), t)\sqrt{\Delta t}z_t$$

Crash Course in Differential Equations

Lecture #15
HY-673

Ordinary Differential Equation (ODE):

$$\frac{dx_t}{dt} = f(x_t, t) \text{ or } dx_t = f(x_t, t)dt$$



Analytical Solution:

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

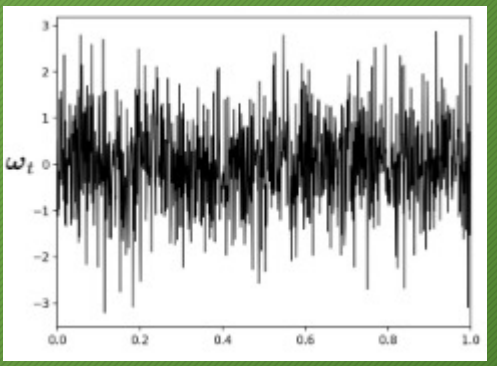
Iterative Numerical Solution:

$$x(t + \Delta t) \approx x(t) + f(x(t), t)\Delta t$$

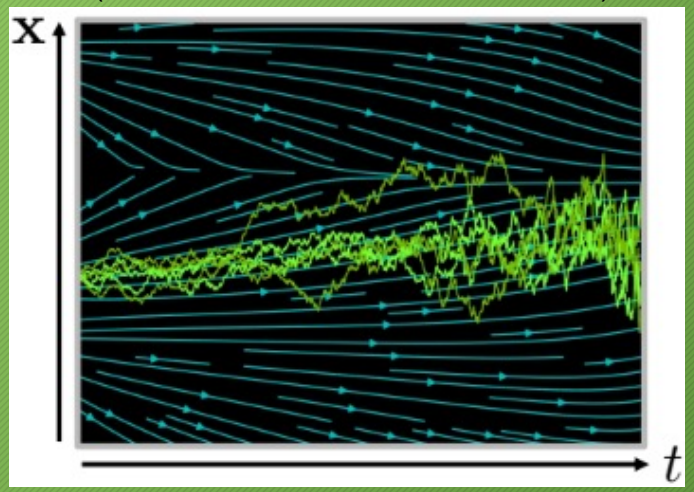
Stochastic Differential Equation (SDE):

$$\frac{dx_t}{dt} = \underbrace{f(x_t, t)}_{\text{drift coefficient}} + \underbrace{\sigma(x_t, t)}_{\text{diffusion coefficient}} W'_t$$

Wiener Process Derivative (i.e., Gaussian White Noise)



$$\left(dx_t = f(x_t, t)dt + \sigma(x_t, t)dW_t\right)$$



$$x(t + \Delta t) \approx x(t) + f(x(t), t)\Delta t + \sigma(x(t), t)\sqrt{\Delta t}z_t$$

Forward Diffusion Process as Stochastic Differential Equation

Lecture #15
HY-673

Consider the limit of **many small steps:**

Forward diffusion process (fixed)

Data  **Noise**

$\mathbf{x}_0$ $\mathbf{x}_1$ $\cdots$ $\cdots$ $\mathbf{x}_T$

$$x_t - x_{t-1} \approx \frac{\beta(t)\Delta(t)}{2} x_{t-1} + \sqrt{\beta(t)\Delta t} z_t$$

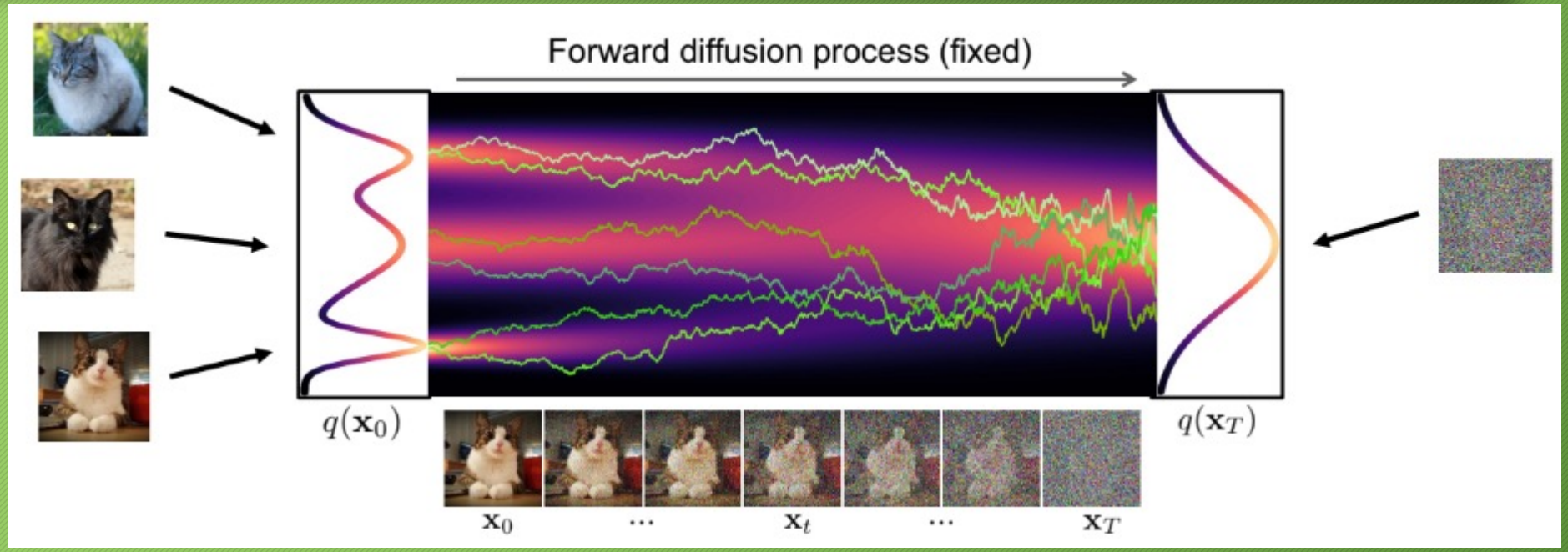
$$dx_t = -\frac{1}{2}\beta(t) x_t dt + \sqrt{\beta(t)} dW_t$$

Stochastic Differential Equation (SDE)
describing the diffusion process in the infinitesimal limit

https://arxiv.org/abs/2011.13456

Forward Diffusion Process as Stochastic Differential Equation

Lecture #15
HY-673



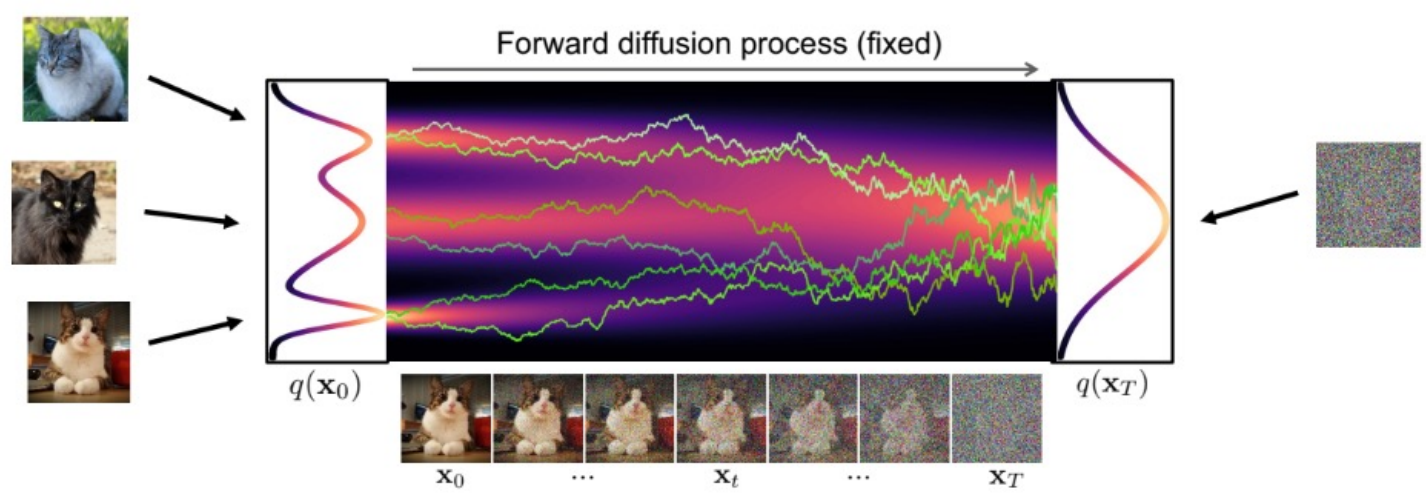
Forward Diffusion SDE:

$$dx_t = \underbrace{-\frac{1}{2}\beta(t)x_t dt}_{\substack{\text{drift term} \\ \text{(pulls towards mode)}}} + \underbrace{\sqrt{\beta(t)}dW_t}_{\substack{\text{diffusion term} \\ \text{(injects noise)}}}.$$

https://arxiv.org/abs/2011.13456

Forward Diffusion Process as Stochastic Differential Equation

Lecture #15
HY-673



Forward Diffusion SDE:

$$dx_t = \underbrace{-\frac{1}{2}\beta(t)x_t dt}_{\substack{\text{drift term} \\ \text{(pulls towards mode)}}} + \underbrace{\sqrt{\beta(t)}dW_t}_{\substack{\text{diffusion term} \\ \text{(injects noise)}}}$$

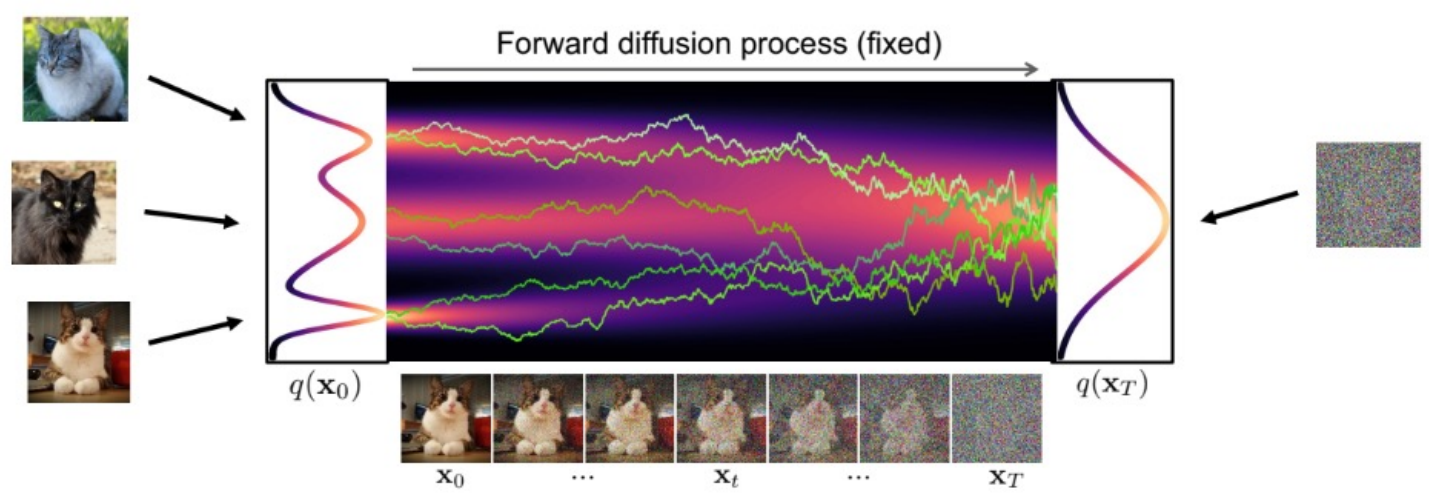
Special case of more general SDEs used in generative diffusion models:

$$dx_t = f(t)x_t dt + g(t)dW_t.$$

https://arxiv.org/abs/2011.13456

The Generative Reverse Stochastic Differential Equation

Lecture #15
HY-673



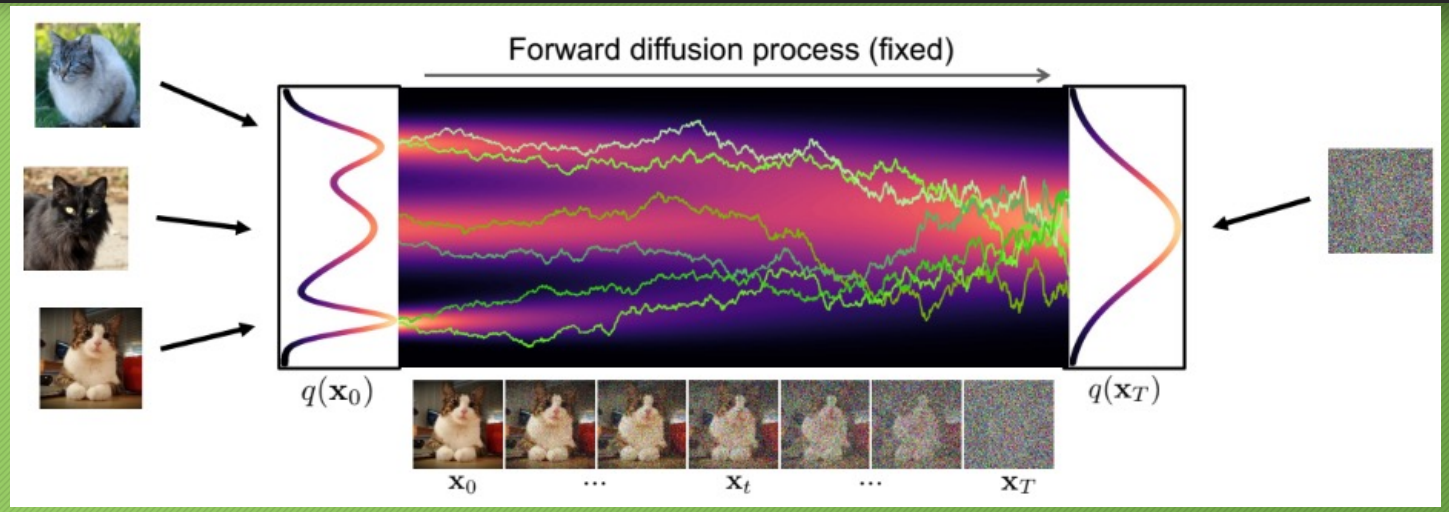
Forward Diffusion SDE: $$dx_t = -\frac{1}{2}\beta(t)x_t dt + \sqrt{\beta(t)}dW_t$$

But what about the reverse process, necessary for generation?

https://arxiv.org/abs/2011.13456

The Generative Reverse Stochastic Differential Equation

Lecture #15
HY-673



Forward Diffusion SDE:

$$dx_t = -\frac{1}{2}\beta(t)x_t dt + \sqrt{\beta(t)}dW_t$$

Reverse Generative Diffusion SDE:

$$dx_t = \overbrace{\left[-\frac{1}{2}\beta(t)x_t - \beta(t)\underbrace{\nabla_x \log q_t(x_t)}_{\text{“Score Function”}}\right]}^{\text{drift term}} dt + \overbrace{\sqrt{\beta(t)}d\bar{W}_t}^{\text{diffusion term}}.$$

https://arxiv.org/abs/2011.13456
https://www.sciencedirect.com/science/article/pii/0304414982900515

➡ Simulate reverse diffusion process: Data generation from random noise!

The Generative Reverse Stochastic Differential Equation

Lecture #15
HY-673

But how to get the score function $\nabla x \log q_t(x_t)$?

Forward Diffusion SDE:

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

Reverse Generative Diffusion SDE:

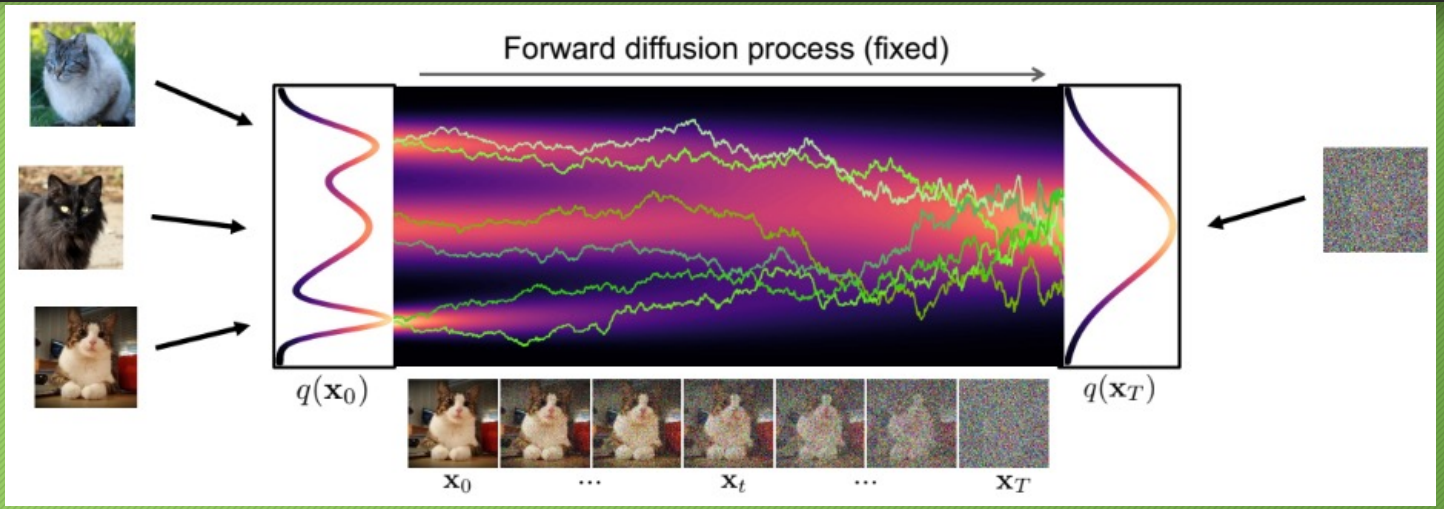
$$d\mathbf{x}_t = \overbrace{\left[-\frac{1}{2}\beta(t)\mathbf{x}_t - \beta(t)\underbrace{\nabla_{\mathbf{x}_t}\log q_t(\mathbf{x}_t)}_{\text{"Score Function"}}\right]}^{\text{drift term}} dt + \overbrace{\sqrt{\beta(t)}d\bar{\boldsymbol{\omega}}_t}^{\text{diffusion term}}$$

https://arxiv.org/abs/2011.13456
https://www.sciencedirect.com/science/article/pii/0304414982900515

➡ Simulate reverse diffusion process: Data generation from random noise!

Score Matching

Lecture #15
HY-673



- Naive idea: learn a model for the score function by direct regression.

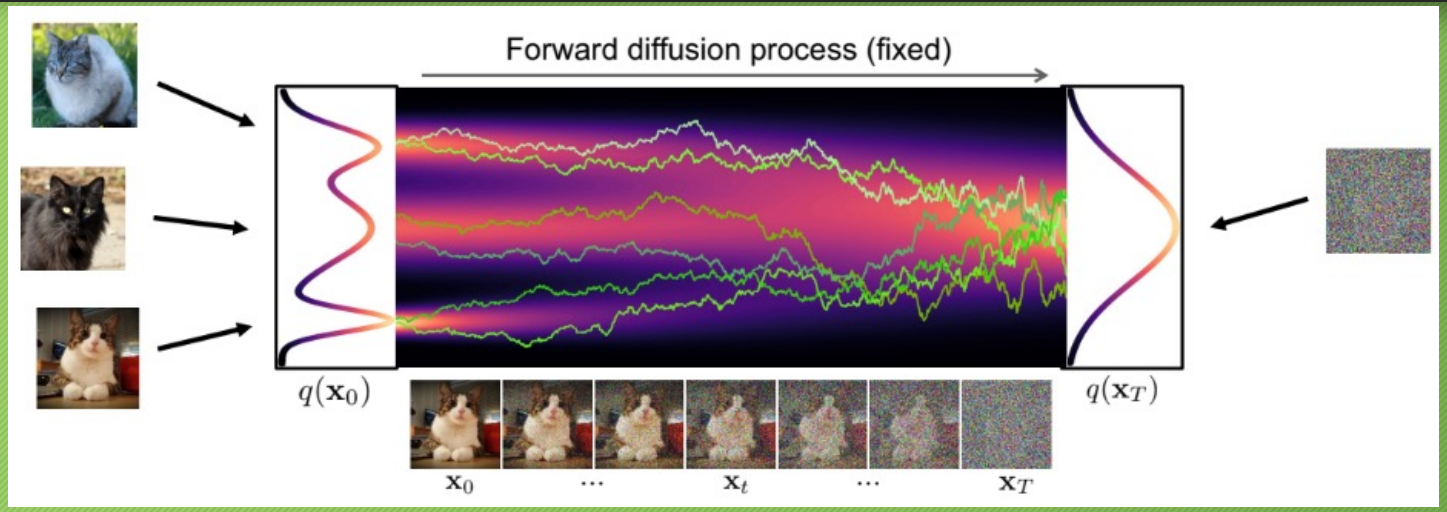
$$\min_\theta \underbrace{\mathbb{E}_{t\sim\mathcal{U}(0,T)}}_{\substack{\text{diffusion}\\ \text{time } t}} \underbrace{\mathbb{E}_{x_t\sim q_t(x_t)}}_{\substack{\text{diffused}\\ \text{data } x_t}} || \underbrace{s_\theta(x_t,t)}_{\substack{\text{neural}\\ \text{network}}} - \underbrace{\nabla x \log q_t(x_t)}_{\substack{\text{score}\\ \text{diffused data}\\ \text{(marginal)}}} ||_2^2.$$

⇒ **But $\nabla_{\mathbf{x}} \log q_t(x_t)$ (score of the marginal diffused density $q_t(x_t)$) is not tractable!**

https://ieeexplore.ieee.org/document/6795935
https://arxiv.org/abs/1907.05600
https://arxiv.org/abs/2011.13456

Denoising Score Matching

Lecture #15
HY-673



"Variance Preserving" SDE:

$$dx_t = -\frac{1}{2}\beta(t)x_t dt + \sqrt{\beta(t)}dW_t$$

$$q_t(x_t|x_0) = \mathcal{N}(x_t; \gamma_t x_0, \sigma_t^2 I)$$

$$\gamma_t = \exp^{-\frac{1}{2}\int_0^t \beta(s)ds}$$

$$\sigma_t^2 = 1 - \exp^{-\frac{1}{2}\int_0^t \beta(s)ds}$$

- Instead, diffuse individual data points x_0. Conditional $q_t(x_t|x_0)$ ***is*** tractable!
- **Denoising Score Matching:**

$$\min_\theta \underbrace{\mathbb{E}_{t\sim\mathcal{U}(0,T)}}_{\substack{\text{diffusion}\\ \text{time } t}} \underbrace{\mathbb{E}_{x_0\sim q_0(x_0)}}_{\substack{\text{diffused}\\ \text{data } x_0}} \underbrace{\mathbb{E}_{x_t\sim q_t(x_t|x_0)}}_{\substack{\text{diffused data}\\ \text{sample } x_t|x_0}} || \underbrace{s_\theta(x_t,t)}_{\substack{\text{neural}\\ \text{network}}} - \underbrace{\nabla x \log q_t(x_t|x_0)}_{\substack{\text{score of diffused}\\ \text{data sample}}} ||_2^2.$$

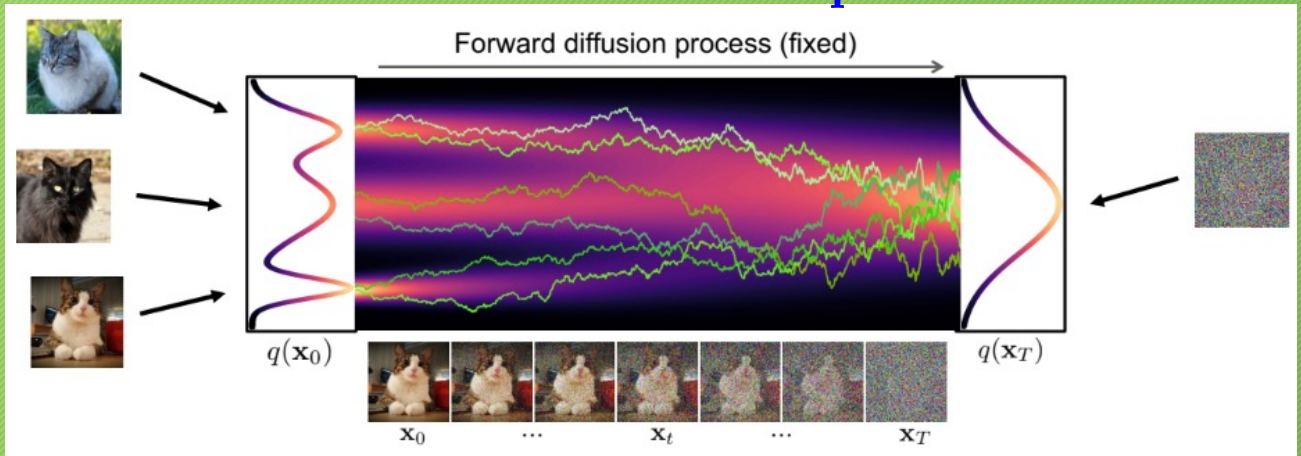
⇒ **After expectations,** $s_\theta(x_t,t) \approx \nabla_x \log q_t(x_t)$**!**

https://ieeexplore.ieee.org/document/6795935
https://arxiv.org/abs/1907.05600
https://arxiv.org/abs/2011.13456

Denoising Score Matching

Lecture #15
HY-673

Implementation 1: Noise Prediction



"Variance Preserving" SDE:

$$dx_t = -\frac{1}{2}\beta(t)x_t dt + \sqrt{\beta(t)}dW_t$$

$$q_t(x_t|x_0) = \mathcal{N}(x_t; \gamma_t x_0, \sigma_t^2 I)$$

$$\gamma_t = \exp^{-\frac{1}{2}\int_0^t \beta(s)ds}$$

$$\sigma_t^2 = 1 - \exp^{-\frac{1}{2}\int_0^t \beta(s)ds}$$

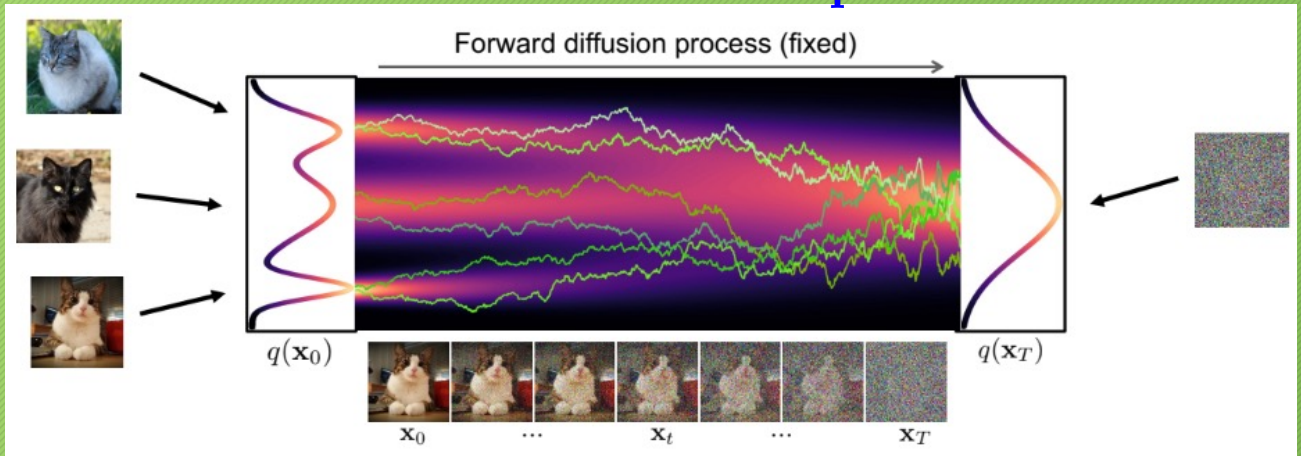
- **Denoising Score Matching:** $\min_\theta \mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{x_0\sim q_0(x_0)}\mathbb{E}_{x_t\sim q_t(x_t|x_0)}||s_\theta(x_t,t) - \nabla x \log q_t(x_t|x_0)||_2^2.$
  - **Re-parametrized Sampling:** $x_t = \gamma_t x_0 + \sigma_t \epsilon, \quad \epsilon \sim \mathcal{N}(0, I).$
  - **Score Function:** $\nabla_x \log q_t(x_t|x_0) = -\nabla_x \frac{(x_t - \gamma_t x_0)^2}{2\sigma_t^2} = -\frac{x_t - \gamma_t x_0}{\sigma_t^2} = -\frac{\gamma_t x_0 + \sigma_t \epsilon - \gamma_t x_0}{\sigma_t^2} = -\frac{\epsilon}{\sigma_t}.$
  - **Neural Network Model:** $\sigma_\theta(x_t, t) := -\frac{\epsilon_\theta(x_t, t)}{\sigma_t}.$

https://ieeexplore.ieee.org/document/6795935
https://arxiv.org/abs/1907.05600
https://arxiv.org/abs/2011.13456

Denoising Score Matching

Lecture #15
HY-673

Implementation 1: Noise Prediction



"Variance Preserving" SDE:

$$dx_t = -\frac{1}{2}\beta(t)x_t dt + \sqrt{\beta(t)}dW_t$$

$$q_t(x_t|x_0) = \mathcal{N}(x_t; \gamma_t x_0, \sigma_t^2 I)$$

$$\gamma_t = \exp^{-\frac{1}{2}\int_0^t \beta(s)ds}$$

$$\sigma_t^2 = 1 - \exp^{-\frac{1}{2}\int_0^t \beta(s)ds}$$

- **Denoising Score Matching:**

$$\min_\theta \mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{x_0\sim q_0(x_0)}\mathbb{E}_{x_t\sim q_t(x_t|x_0)}||s_\theta(x_t,t) - \nabla x \log q_t(x_t|x_0)||_2^2.$$

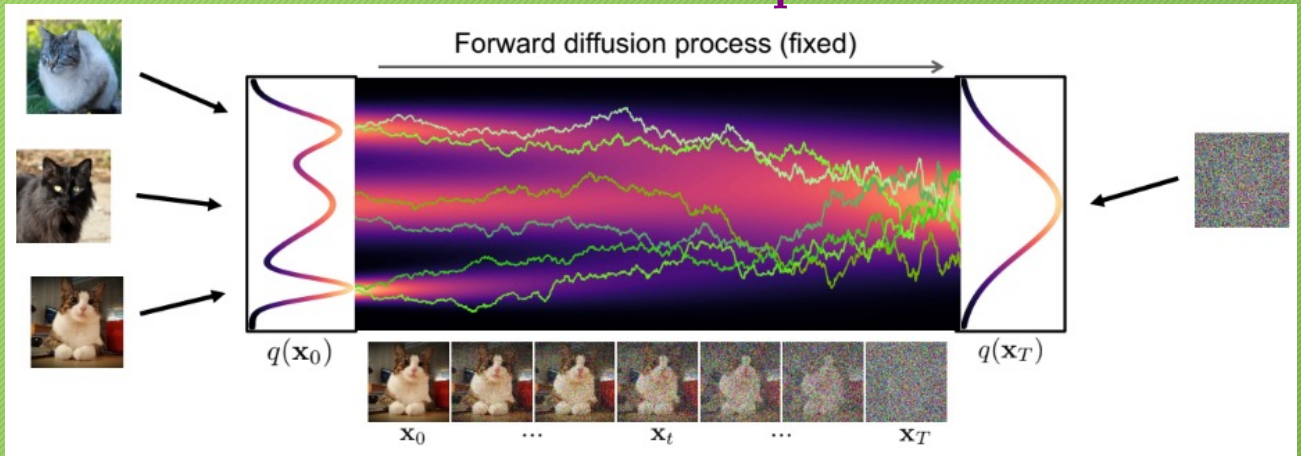
$$\Rightarrow \quad \min_\theta \mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{x_0\sim q_0(x_0)}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}\frac{1}{\sigma_t^2}||\epsilon - \epsilon_\theta(x_t,t)||_2^2.$$

https://ieeexplore.ieee.org/document/6795935
https://arxiv.org/abs/1907.05600
https://arxiv.org/abs/2011.13456

Denoising Score Matching

Lecture #15
HY-673

Implementation 2: Loss Weightings



"Variance Preserving" SDE:

$$dx_t = -\frac{1}{2}\beta(t)x_t dt + \sqrt{\beta(t)}dW_t$$

$$q_t(x_t|x_0) = \mathcal{N}(x_t; \gamma_t x_0, \sigma_t^2 I)$$

$$\gamma_t = \exp^{-\frac{1}{2}\int_0^t \beta(s)ds}$$

$$\sigma_t^2 = 1 - \exp^{-\frac{1}{2}\int_0^t \beta(s)ds}$$

- **Denoising Score Matching** objective with loss weighting $\lambda(t)$:

$$\min_\theta \mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{x_0\sim q_0(x_0)}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}\frac{\lambda(t)}{\sigma_t^2}||\epsilon - \epsilon_\theta(x_t,t)||_2^2.$$

Different loss weighting trade off between model with good quality vs. high log-likelihood:

- Perceptual quality: $\lambda(t) = \sigma_t^2$
- Maximum log-likelihood: $\lambda(t)\beta(t)$ *(negative ELBO)*

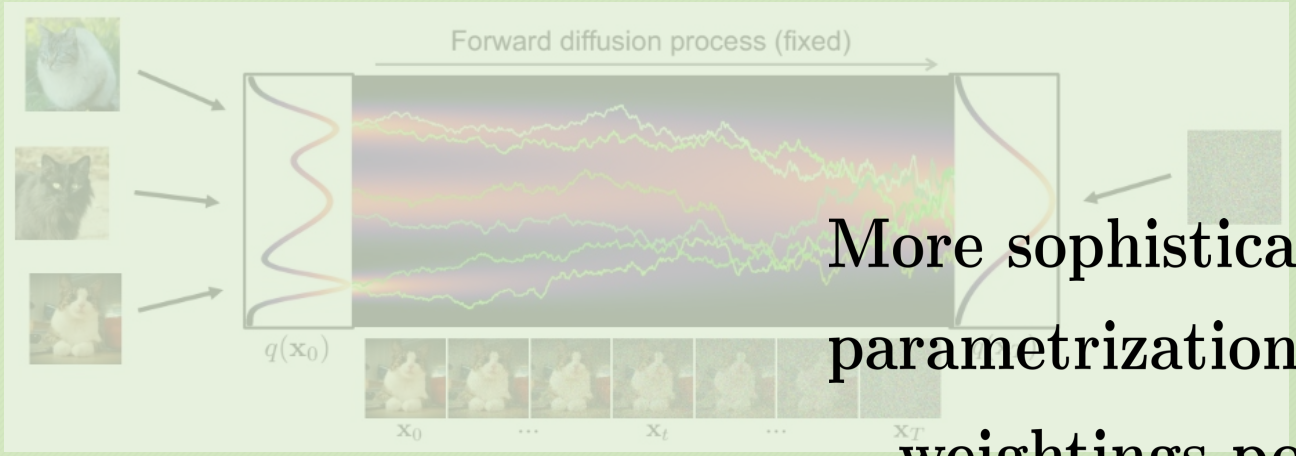
⇒ **Same objectives as derived in Denoising DPMs!**

https://ieeexplore.ieee.org/document/6795935
https://arxiv.org/abs/1907.05600
https://arxiv.org/abs/2011.13456

Denoising Score Matching

Lecture #15
HY-673

Implementation 2: Loss Weightings



"Variance Preserving" SDE:

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

$$q_t(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \gamma_t \mathbf{x}_0, \sigma_t^2 \mathbf{I})$$

$$\gamma_t = \exp^{-\frac{1}{2}\int_0^t \beta(s)ds}$$

$$\sigma_t^2 = 1 - \exp^{-\frac{1}{2}\int_0^t \beta(s)ds}$$

More sophisticated model parametrizations and loss weightings possible!

- **Denoising Score Matching** objective with loss weighting $\lambda(t)$: $\min_{\theta} \mathbb{E}_{t\sim\mathcal{U}(0,\mathrm{T})}\mathbb{E}_{\mathbf{x}_0\sim q_0(\mathbf{x}_0)}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\frac{\lambda(t)}{\sigma_t^2}||\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)||_2^2.$

Different loss weighting trade off between model with good quality vs. high log-likelihood

- Perceptual quality: $\lambda(t) = \sigma_t^2$
- Maximum log-likelihood: $\lambda(t)\beta(t)$ *(negative ELBO)*

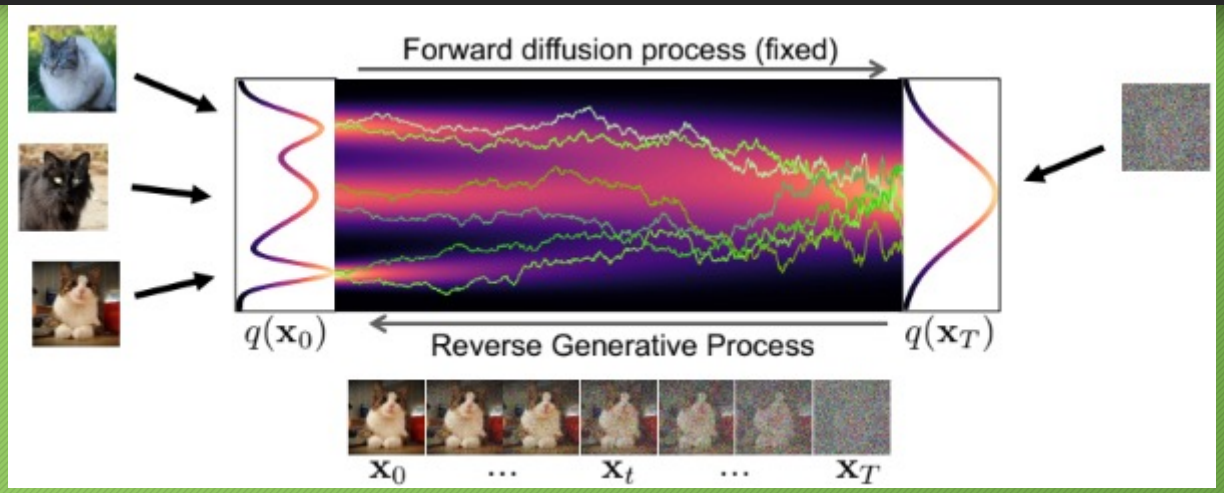
➡ Karras et al., "Elucidating the Design Space of Diffusion-Based Generative Models"

➡ Same objectives as derived with variational approach in Part (1)!

https://arxiv.org/abs/2006.11239
https://arxiv.org/abs/2101.09258
https://arxiv.org/abs/2107.00630
https://arxiv.org/abs/2106.05931
https://arxiv.org/abs/2106.02808
https://arxiv.org/abs/2206.00364

Probability Flow ODE

Lecture #15
HY-673



- Consider reverse generative diffusion SDE:

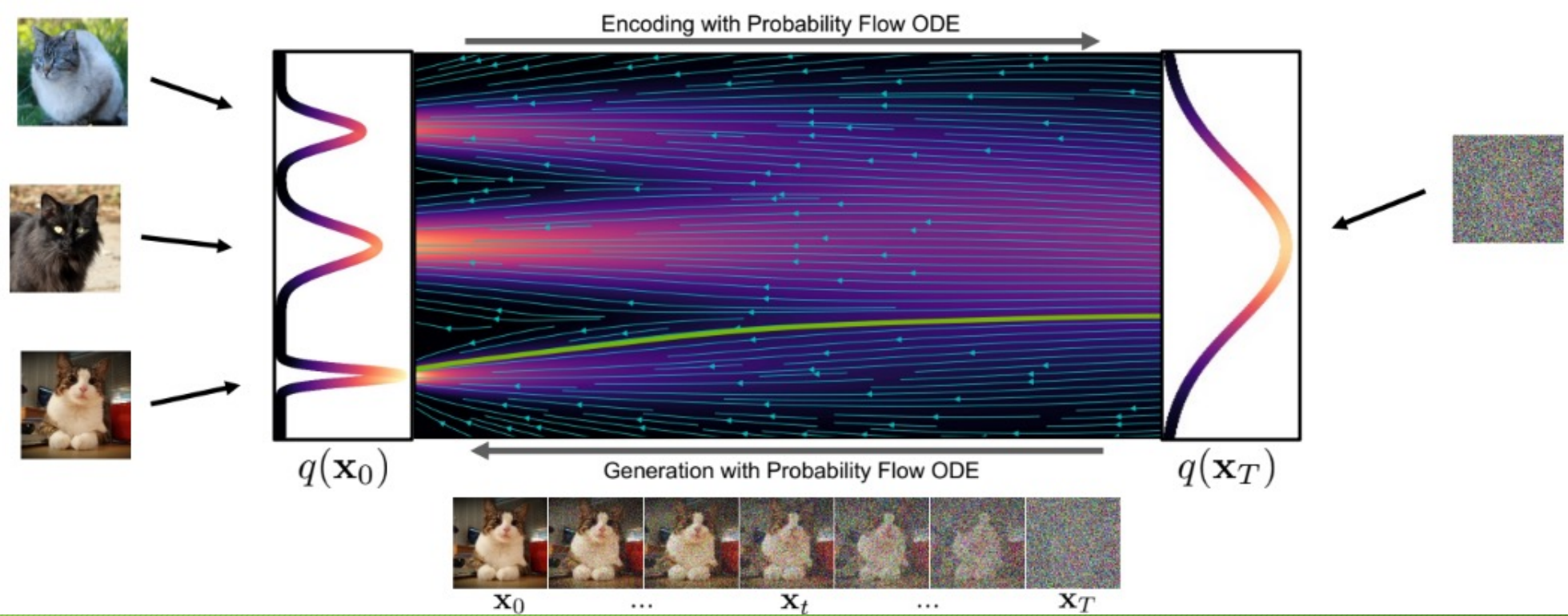
$$dx_t = -\frac{1}{2}\beta(t)\Big[x_t + 2\nabla_x \log q_t(x_t)\Big]dt + \sqrt{\beta(t)}d\bar{W}_t.$$

- Equivalent to **"Probability Flow ODE"** in distribution:
  (i.e., solving this ODE results in the same $q_t(x_t)$ when initializing $q_T(x_T) \approx \mathcal{N}(x_T; 0, I)$)

$$dx_t = -\frac{1}{2}\beta(t)\Big[x_t + \nabla_x \log q_t(x_t)\Big]dt.$$

https://arxiv.org/abs/2011.13456

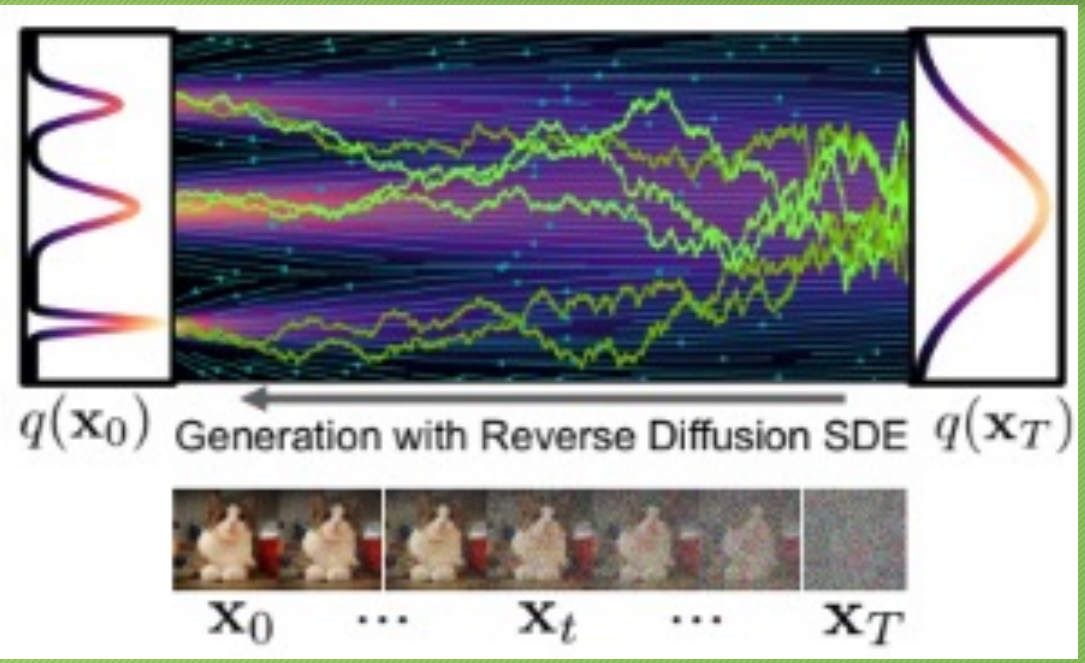
Probability Flow ODE

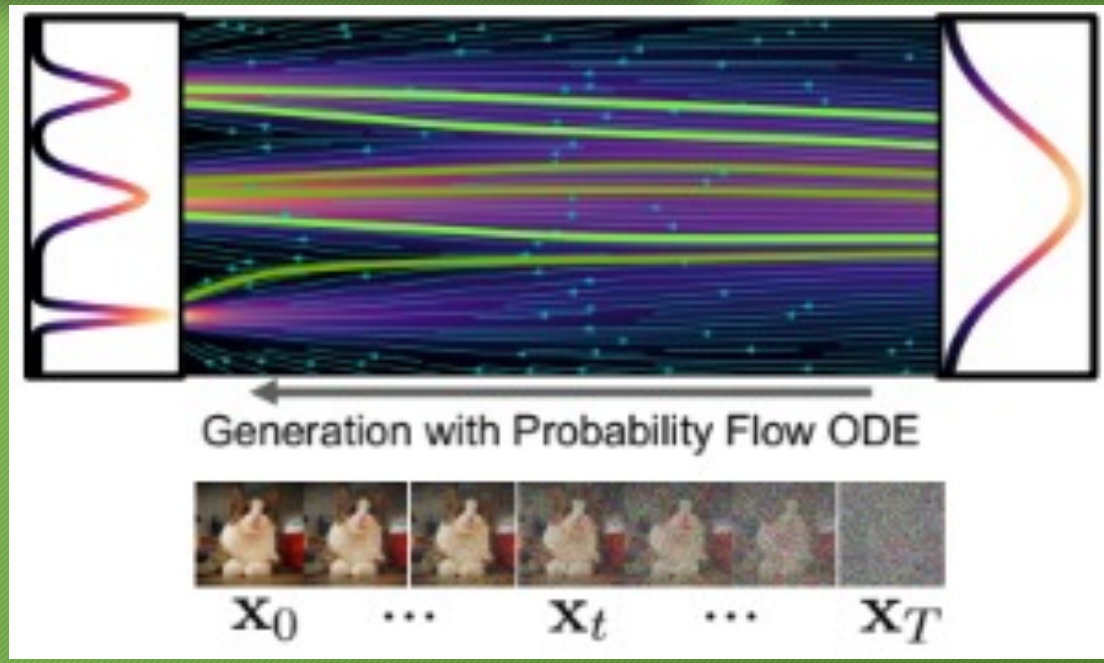


$$dx_t = -\frac{1}{2}\beta(t)\Big[x_t + s_\theta(x_t)\Big]dt.$$

https://arxiv.org/abs/2011.13456

Synthesis with SDE vs. ODE





- **Generative Reverse Diffusion SDE (stochastic):**

$$dx_t = -\frac{1}{2}\beta(t)\Big[x_t + 2s_\theta(x_t)\Big]dt + \sqrt{\beta(t)}d\bar{W}_t.$$

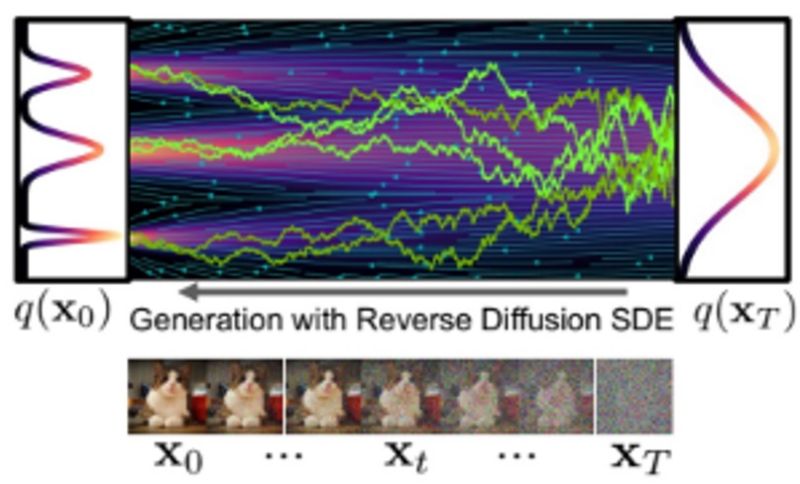
- **Generative Probability Flow ODE (deterministic):**

$$dx_t = -\frac{1}{2}\beta(t)\Big[x_t + s_\theta(x_t)\Big]dt.$$

https://arxiv.org/abs/2011.13456

Sampling from "Continuous-Time" Diffusion Models

Lecture #15
HY-673



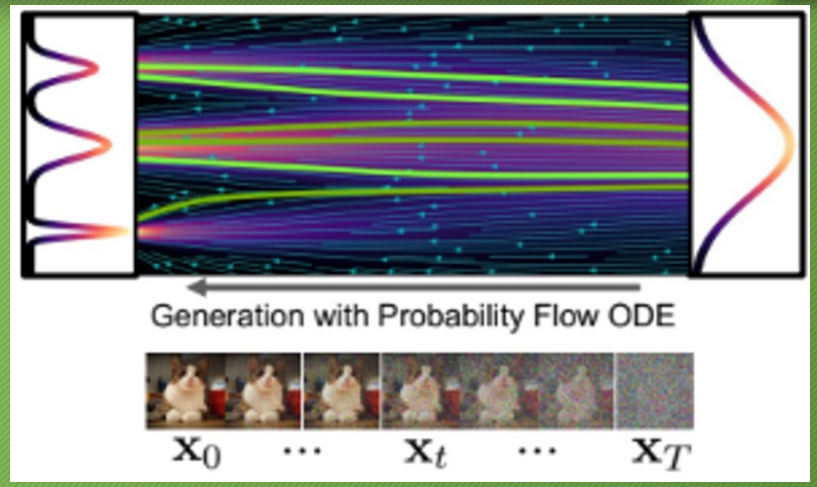
- **Generative (Reverse) Diffusion SDE:**

$$dx_t = -\frac{1}{2}\beta(t)\Big[x_t + 2s_\theta(\mathbf{x}_t)\Big]dt + \sqrt{\beta(t)}d\bar{W}_t.$$

1. **Euler-Maruyama:**

$$x_{t-1} = x_t + \frac{1}{2}\beta(t)\Big[x_t + 2s_\theta(x_t)\Big]\Delta t + \sqrt{\beta(t)\Delta t}\bar{z}_t.$$

2. **Ancestral Sampling is also a generative SDE sampler!**



- **Probability Flow ODE:**

$$dx_t = -\frac{1}{2}\beta(t)\Big[x_t + s_\theta(x_t)\Big]dt.$$

1. **Euler's Method:**

$$x_{t-1} = x_t + \frac{1}{2}\beta(t)\Big[x_t + s_\theta(x_t)\Big]\Delta t.$$

2. **In practice: Higher-Order ODE solvers (Runge-Kutta, linear multistep methods, exponential integrators).**

https://arxiv.org/abs/2011.13456

Sampling from "Continuous-Time" Diffusion Models

Lecture #15
HY-673

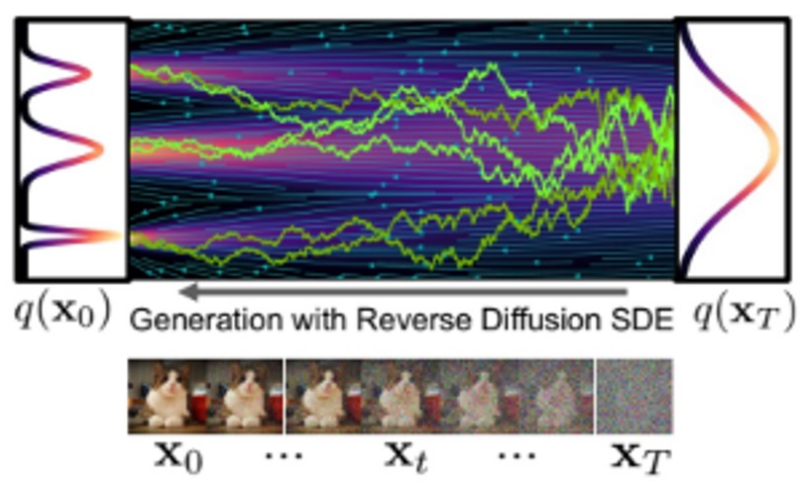
How to solve the generative SDE or ODE in practice?

- Runge-Kutta adaptive step-size ODE solver [1]
- Higher-Order adaptive step-size SDE solver [2]
- Reparametrized, smoother ODE [3]
- Higher-Order ODE soolver with linear multistepping [4]
- Exponential ODE Integrators [5, 6]
- Higher-Order ODE solver with Heun's Method [7]

[1] https://arxiv.org/abs/2011.13456
[2] https://arxiv.org/abs/2105.14080
[3] https://arxiv.org/abs/2010.02502
[4] https://arxiv.org/abs/2202.09778
[5] https://arxiv.org/abs/2204.13902
[6] https://arxiv.org/abs/2206.00927
[7] https://arxiv.org/abs/2206.00364

Sampling from "Continuous-Time" Diffusion Models

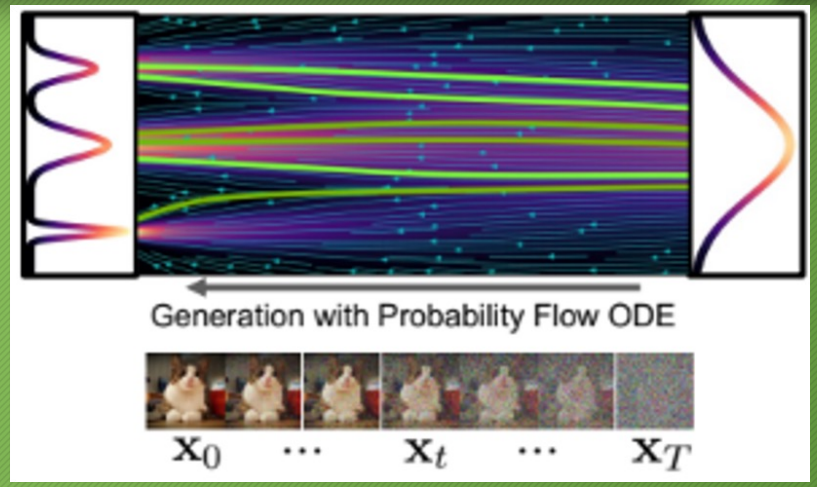
Lecture #15
HY-673



- **Generative (Reverse) Diffusion SDE:**

$$dx_t = \underbrace{-\frac{1}{2}\beta(t)[x_t + s_\theta(x_t)]dt}_{\text{Probability Flow ODE}} \underbrace{-\frac{1}{2} + \beta(t)[s_\theta(x_t)]dt\sqrt{\beta(t)}d\bar{W}_t}_{\text{Langevin Diffusion}}.$$

1. **Pros:** Continuous noise injection can help to compensate errors during diffusion process (Langevin sampling acttively pushes towards correct distribution).
2. **Cons:** Often slower, beacuse the stochastic terms themselves require fine discretization during solve.



- **Probability Flow ODE:**

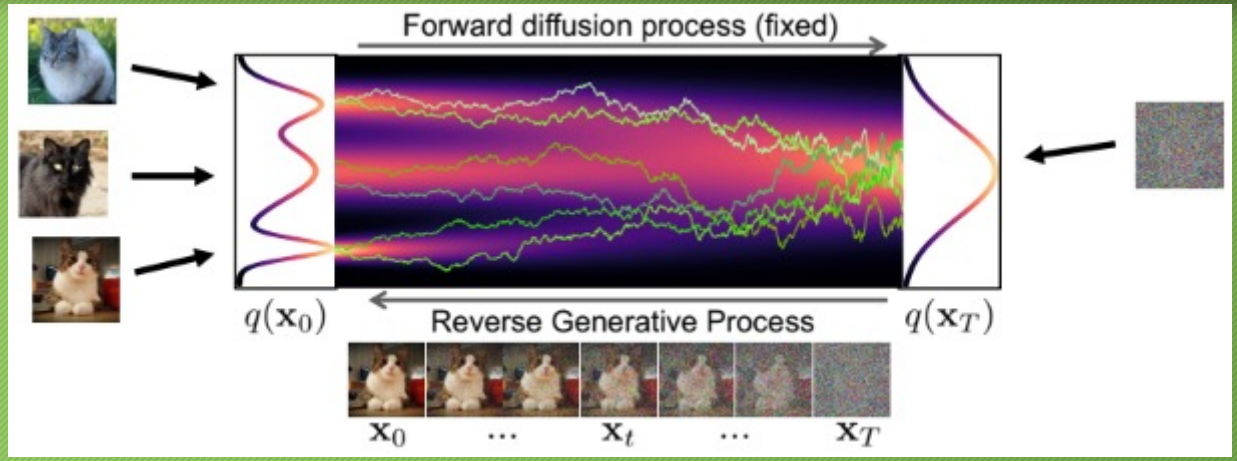
$$dx_t = -\frac{1}{2}\beta(t)\Big[x_t + s_\theta(x_t)\Big]dt.$$

1. **Pros:** Can leverage fast ODE solvers. Best when targeting very fast sampling.
2. **Cons:** No "stochastic" error correction, often slightly lower performance than stochastic sampling.

https://arxiv.org/abs/2206.00364

Why use Differential Equation Framework?

Lecture #15
HY-673

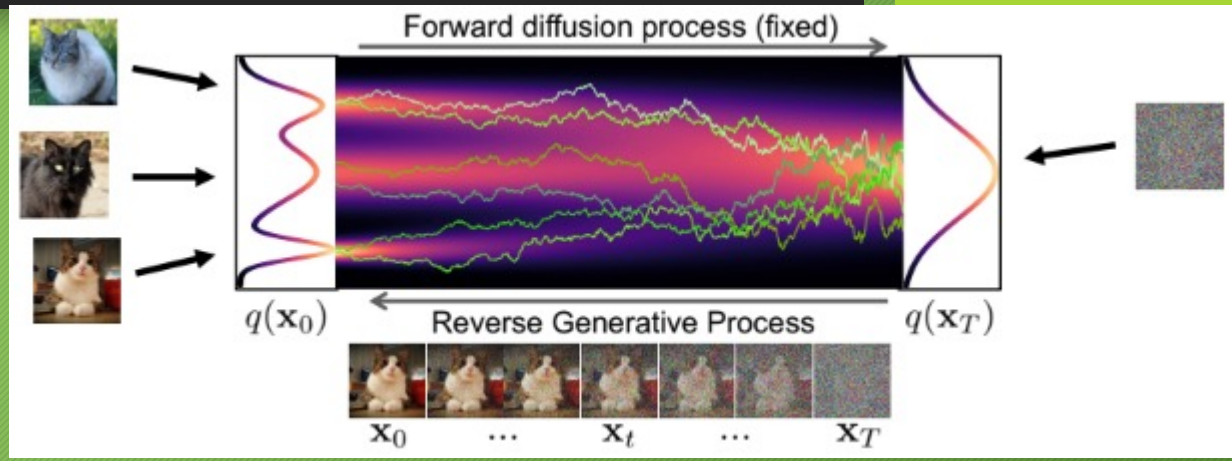


Advantages of the Differential Equation framework for Diffusion Models:

- Can leverage broad existing literature on **advanced and fast SDE and ODE solvers**.
- Allows us to construct deterministic **Probability Flow ODE**.
  - Deterministic Data Encodings.
  - Log-likelihood Estimation.
- **Clean mathematical framework** based on Diffusion Processes and Score Matching; connections to Neural ODEs, Continuous Normalizing Flows and Energy-based Models.

Diffusion Models as Energy-based Models

Lecture #15
HY-673



- Assume an Energy-base Model (EBM): $p_\theta(x,t) = \dfrac{e^{-E_\theta(x,t)}}{Z_{\theta,t}}.$
- Sample EBM via Langevin dynamics: $x_{t-1} = x_t - \eta \nabla_x E_\theta(x_t,t) + \sqrt{2\eta} z_t.$
- Requires only gradient of energy $-\nabla_x E(x,t)$, not $E_\theta(x,t)$ itself, nor $Z_{\theta,t}$!

In diffusion models, we learn "energy gradients" for all diffused distributions directly:

$$\nabla_x \log q_t(x) \approx s_\theta(x,t) =: \nabla_x \log p_\theta(x,t) = -\nabla_x E_\theta(x,t) - \underbrace{\nabla_x \log Z_{\theta,t}}_{=0} = -\nabla_x E_\theta(x,t).$$

Diffusion Models model energy gradient directly, along entire diffusion process, and avoid modeling partition function. Different noise levels along diffusion are analogous to annealed sampling in EBMs.

Introduction to Deep Generative Modeling

Lecture #15

HY-673 – Computer Science Dep, University of Crete

Professors: Yannis Pantazis, Yannis Stylianou

Teaching Assistant: Thomas Marchioro