MotionDiffuse: Text-Driven Human Motion Generation with Diffusion Model

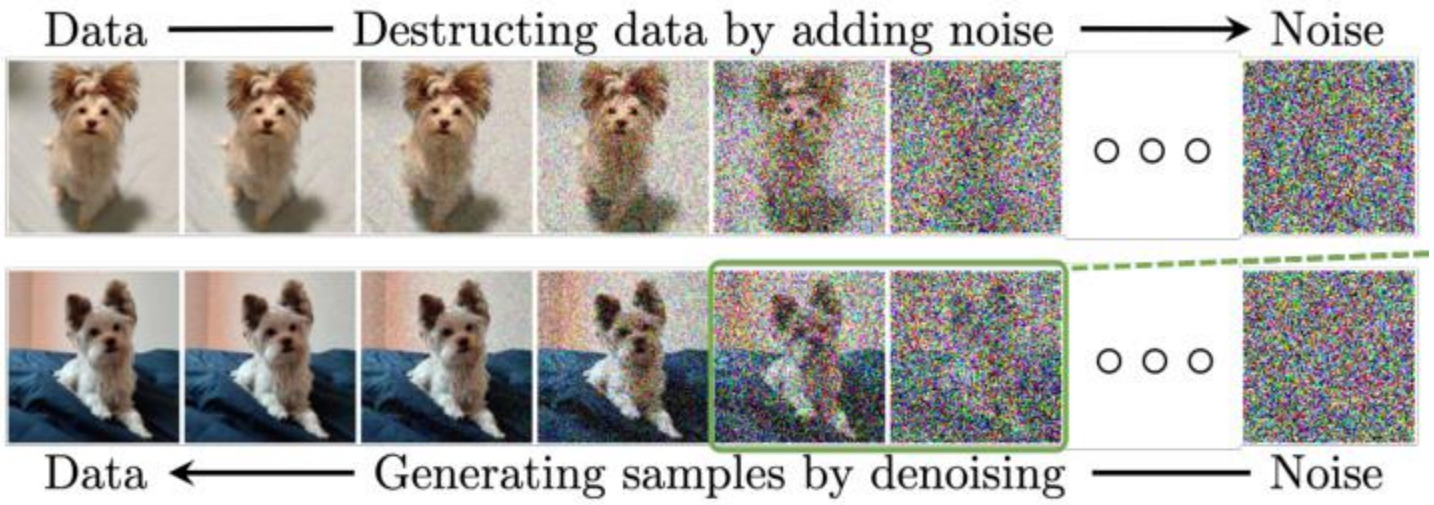
Mingyuan Zhang Zhongang Cai Liang Pan Fangzhou Hong Xinying Guo Lei Yang
Ziwei Liu

S-Lab, Nanyang Technological University SenseTime Research

Background: Diffusion Models – Overview

- Generative Model capable of state-of-the-art generation of pictures, videos, 3D models, etc.

- Consists of a *forward process*, where a datum is progressively noised, and a *backward process*, where the datum is restored from noise.



Yang, Ling, et al. "Diffusion models: A comprehensive survey of methods and applications.“ *arXiv preprint arXiv:2209.00796* (2022).

Background: Markov Process

- Stochastic Process: sequence of random variables $X_1,\ X_2,\ X_3, \ldots$

- Bernoulli Process, the simplest stochastic process: A sequence of indepe ndent Bernoulli trials $X_i \sim Bernoulli(p)$

  - At each trial, i:

$$\mathrm{P}(X_i = 1) = \mathrm{P}(\text{success at the } i\text{th trial}) = p$$
$$\mathrm{P}(X_i = 0) = \mathrm{P}(\text{failure at the } i\text{th trial}) = 1 - p$$

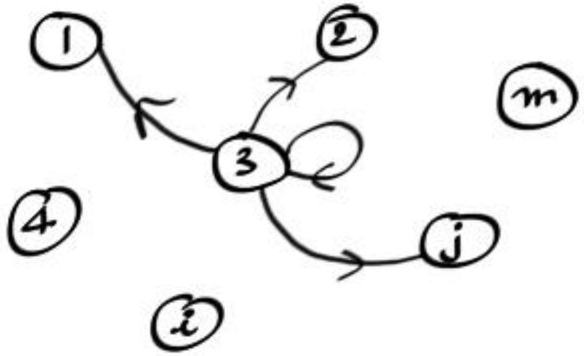
  - Key assumptions:
    – Independence
    – Time-homogeneity
  - Model of:
    – Sequence of lottery wins/losses
    – Arrivals (each second) to a bank
    – Arrivals (at each time slot) to server

https://ocw.mit.edu/courses/res-6-012-introduction-to-probability-spring-2018/pages/part-iii-random-processes/

Background: Markov Process

- Markov Process: Stochastic process satisfying the Markov assumptions
  - X_n: state after n transitions
    - belongs to a finite set
    - initial state X_0 either given or random
    - transition probabilities:

$$p_{ij} = P(X_1 = j \mid X_0 = i) = P(X_{n+1} = j \mid X_n = i)$$ i)



  - Markov property/assumption:
    "given current state, the past doesn't matter"

$$p_{ij} = P(X_{n+1} = j \mid X_n = i) = P(X_{n+1} = j \mid X_n = i,\ X_{n-1}, \ldots, X_0)$$

  - model specification: identify states, transitions, and transition probabilities

https://ocw.mit.edu/courses/res-6-012-introduction-to-probability-spring-2018/pages/part-iii-random-processes/

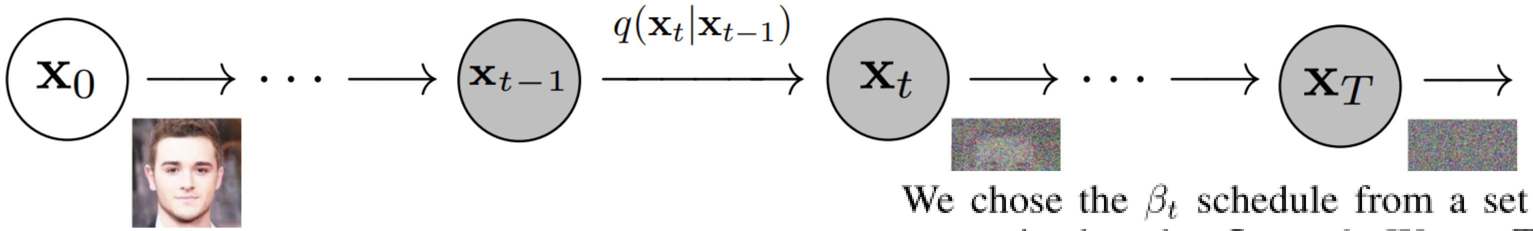
i) More specifically,
• the conditional distribution of Xn given X1, ..., Xn−1 is the same as the conditional distribution of Xn given Xn−1 only, and
• the conditional distribution of Xn given Xn−1 does not depend on n.
(https://www.stat.umn.edu/geyer/f05/8931/n1998.pdf)

Background: Diffusion Models

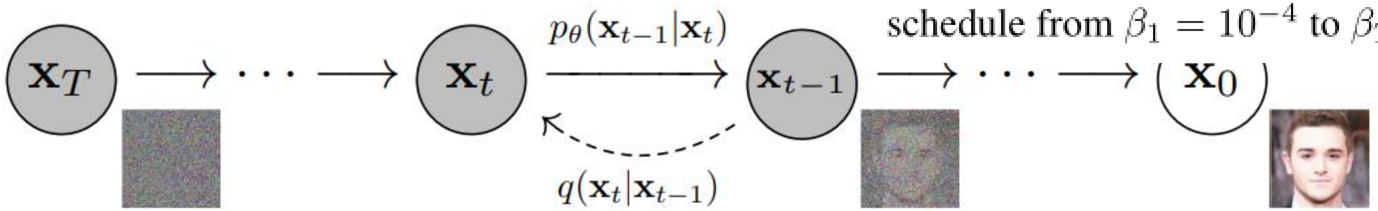
- Will only discuss Denoising Diffusion Probabilistic Models (DDPMs), even though there are more variations (e.g., SGMs, SDEs)
- Diffusion Model maps data $X_0 \sim q(X_0)$ to latent space using a fixed Markov chain to generate progressively-noised random variables $X_0, X_1, X_2, \ldots, X_T$ with transition kernel (i.e., distribution of transition probability) :

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I})$$

where $0 \leq \beta_t \leq 1$ $(t = 0, 1, \ldots, T)$ are hyperparameters.



We chose the β_t schedule from a set of constant, linear, and quadratic schedules, all constrained so that $L_T \approx 0$. We set $T = 1000$ without a sweep, and we chose a linear schedule from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$.



Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." *Advances in Neural Information Processing Systems* 33 (2020): 6840-6851.

https://www.assemblyai.com/blog/diffusion-models-for-machine-learning-introduction/

Background: DDPM – Forward Process

- The *forward process*: $q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I})$

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \prod_{t=1}^{T} \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I})$$

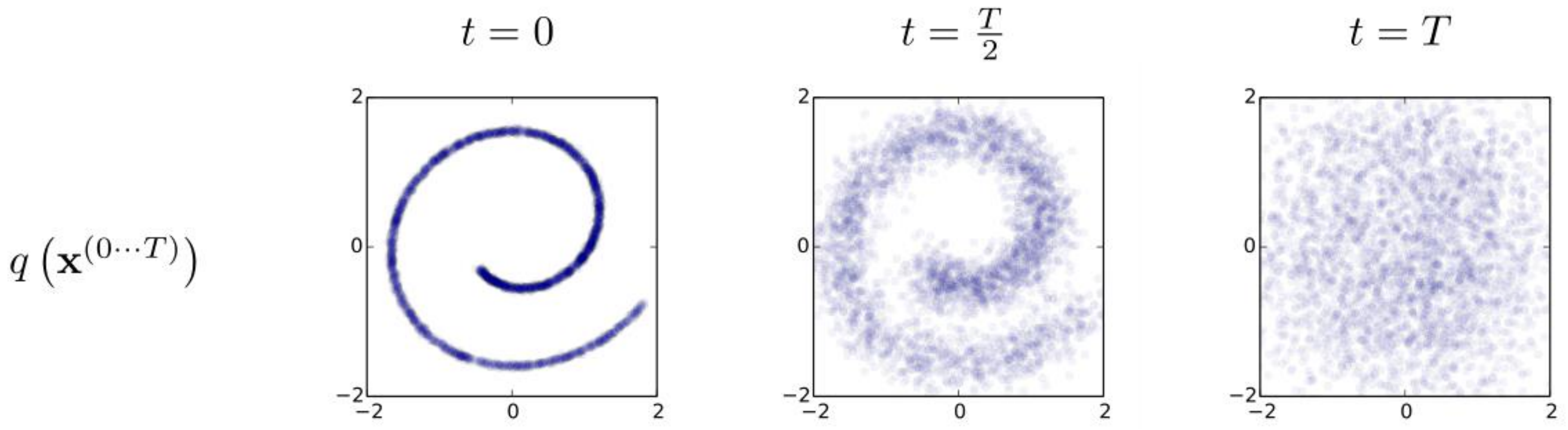
$\sqrt{1-\beta_t} \approx 1$

$\Rightarrow q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$ $\quad (\alpha_t := 1 - \beta_t,\ \bar{\alpha}_t := \prod_{s=0}^{t} \alpha_s)$

progressively adds noise, i.e., $\quad \mathcal{X}_t \sim \mathcal{N}(\mathcal{X}_{t-1}, 1) \iff \mathcal{X}_t = \mathcal{X}_{t-1} + N(0,1)$

until $q(\mathbf{x}_T) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$

Proof can be found in:
https://www.assemblyai.com/blog/diffusion-models-for-machine-learning-introduction/

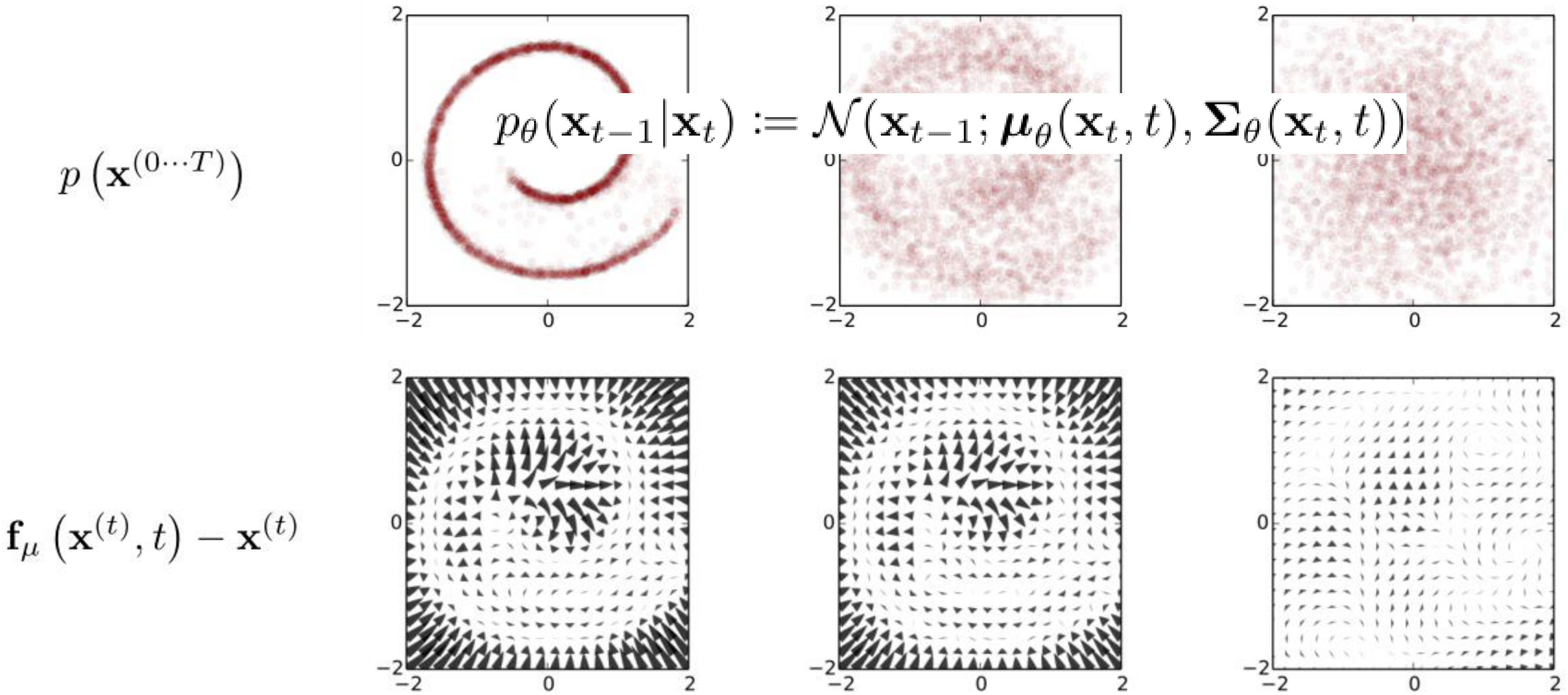


Sohl-Dickstein, Jascha, et al. "Deep unsupervised learning using nonequilibrium thermodynamics."
International Conference on Machine Learning. PMLR, 2015.

Background: DDPM – Reverse Process

- The *reverse process* transforms unit Gaussian noise back to original data by traversing the path backwards using a learnable kernel:

$$p_\theta(\mathbf{x}_{0:T}) := p(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t), \qquad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$



$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

←Drift term during reverse process

Sohl-Dickstein, Jascha, et al. "Deep unsupervised learning using nonequilibrium thermodynamics.“ *International Conference on Machine Learning*. PMLR, 2015.

Background: DDPM – Loss

- Learn to match the joint distribution of the reverse Markov chain $p_\theta(\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_T) := p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ and the forward Markov chain, $q(\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_T) := q(\mathbf{x}_0) \prod_{t=1}^{T} q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$, i.e., minimise KL divergence btw. them:

$$
\begin{aligned}
&\mathrm{KL}(q(\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_T) \,||\, p_\theta(\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_T)) \\
\overset{(i)}{=}\; & -\mathbb{E}_{q(\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_T)}[\log p_\theta(\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_T)] + \text{const} \\
\overset{(ii)}{=}\; & \underbrace{\mathbb{E}_{q(\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_T)}\left[-\log p(\mathbf{x}_T) - \sum_{t=1}^{T} \log \frac{p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)}{q(\mathbf{x}_t \mid \mathbf{x}_{t-1})} \right]}_{:=-L_{\mathrm{VLB}}(\mathbf{x}_0)} + \text{const} \\
\overset{(iii)}{\geq}\; & \mathbb{E}[-\log p_\theta(\mathbf{x}_0)] + \text{const},
\end{aligned}
$$

- (ii) is also the variational lower bound on negative log likelihood (Jensen's Inequality)

Yang, Ling, et al. "Diffusion models: A comprehensive survey of methods and applications." *arXiv preprint arXiv:2209.00796* (2022).

Background: DDPM – Loss

- It is possible to refactor the loss term into a sum of KL divergences:

$$L_{vlb} = L_0 + L_1 + ... + L_{T-1} + L_T$$

where

$$L_0 = -\log p_\theta(x_0|x_1) \quad \leftarrow \text{NLL}$$

$$L_{t-1} = D_{KL}(q(x_{t-1}|x_t, x_0)\;||\;p_\theta(x_{t-1}|x_t))$$

$$L_T = D_{KL}(q(x_T|x_0)\;||\;p(x_T)) \quad \bigstar$$

and since KL div btw. Gaussians have closed-form expressions, they can be exactly calculated.

- If we parametrise the reverse transition kernel as:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I})$$

$$\boldsymbol{\Sigma}_\theta(x_t, t) = \sigma_t^2\mathbb{I}$$

$$\sigma_t^2 = \beta_t$$

Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." *Advances in Neural Information Processing Systems* 33 (2020): 6840-6851. and https://www.assemblyai.com/blog/diffusion-models-for-machine-learning-introduction - For proof of ★ see sup A or Appendix A of this presentation

Background: DDPM – Loss

- Then we have $L_{t-1} = D_{KL}(q(x_{t-1}|x_t, x_0) \,||\, p_\theta(x_{t-1}|x_t))$

$$\Longrightarrow \quad L_{t-1} \propto ||\tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t)||^2$$

and
$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \beta_t \mathbf{I}),$$

$$\text{where} \quad \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \quad \text{and} \quad \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

$$\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}) = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon} \text{ for } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

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

Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." *Advances in Neural Information Processing Systems* 33 (2020): 6840-6851.
and https://www.assemblyai.com/blog/diffusion-models-for-machine-learning-introduction/

Background: DDPM – Loss

$$L_{t-1} = \mathbb{E}_q \left[\frac{1}{2\sigma_t^2} \|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t)\|^2 \right] + C \tag{8}$$

$$L_{t-1} - C = \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}} \left[\frac{1}{2\sigma_t^2} \left\| \tilde{\boldsymbol{\mu}}_t \left(\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}), \frac{1}{\sqrt{\bar{\alpha}_t}} (\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}) - \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}) \right) - \boldsymbol{\mu}_\theta(\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}), t) \right\|^2 \right] \tag{9}$$

$$= \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}} \left[\frac{1}{2\sigma_t^2} \left\| \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}) - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon} \right) - \boldsymbol{\mu}_\theta(\mathbf{x}_t(\mathbf{x}_0, \boldsymbol{\epsilon}), t) \right\|^2 \right] \tag{10}$$

$$\Rightarrow \quad \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}} \left[\frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1 - \bar{\alpha}_t)} \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}, t) \right\|^2 \right] \tag{12}$$

Equation (10) reveals that $\boldsymbol{\mu}_\theta$ must predict $\frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon} \right)$ given $\mathbf{x}_t$. Since $\mathbf{x}_t$ is available as input to the model, we may choose the parameterization

$$\frac{1}{\sqrt{\alpha_t}} \approx 1$$

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

$$\boxed{\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \tilde{\boldsymbol{\mu}}_t \left(\mathbf{x}_t, \frac{1}{\sqrt{\bar{\alpha}_t}} (\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t)) \right) = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) \quad (11)}$$

Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." *Advances in Neural Information Processing Systems* 33 (2020): 6840-6851.

Background: DDPM – Loss

- In conclusion, the loss term reduces to:

$$L_{\text{simple}}(\theta) := \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\underbrace{\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}}_{\mathbf{x}_t \text{ (Slide 10)}}, t)\right\|^2\right] \tag{14}$$

(The intuition is that any data point x_0 eventually turn into part of uniform noise $N(0, I)$ at step T and starting from a point, say y_T we want to find our way back to y_0, which is possible by minimising the joint distribution on Slide 8—can think of it as matching the "path"—but also means that even though every datum gets mapped to a uniform distribution two data must not map to a single point at T because then it will be impossible to trace back in the right path?)

Background: DDPM – Training and Sampling

Shifting "previous" step output x_t by predicted noise (based on x_t)

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

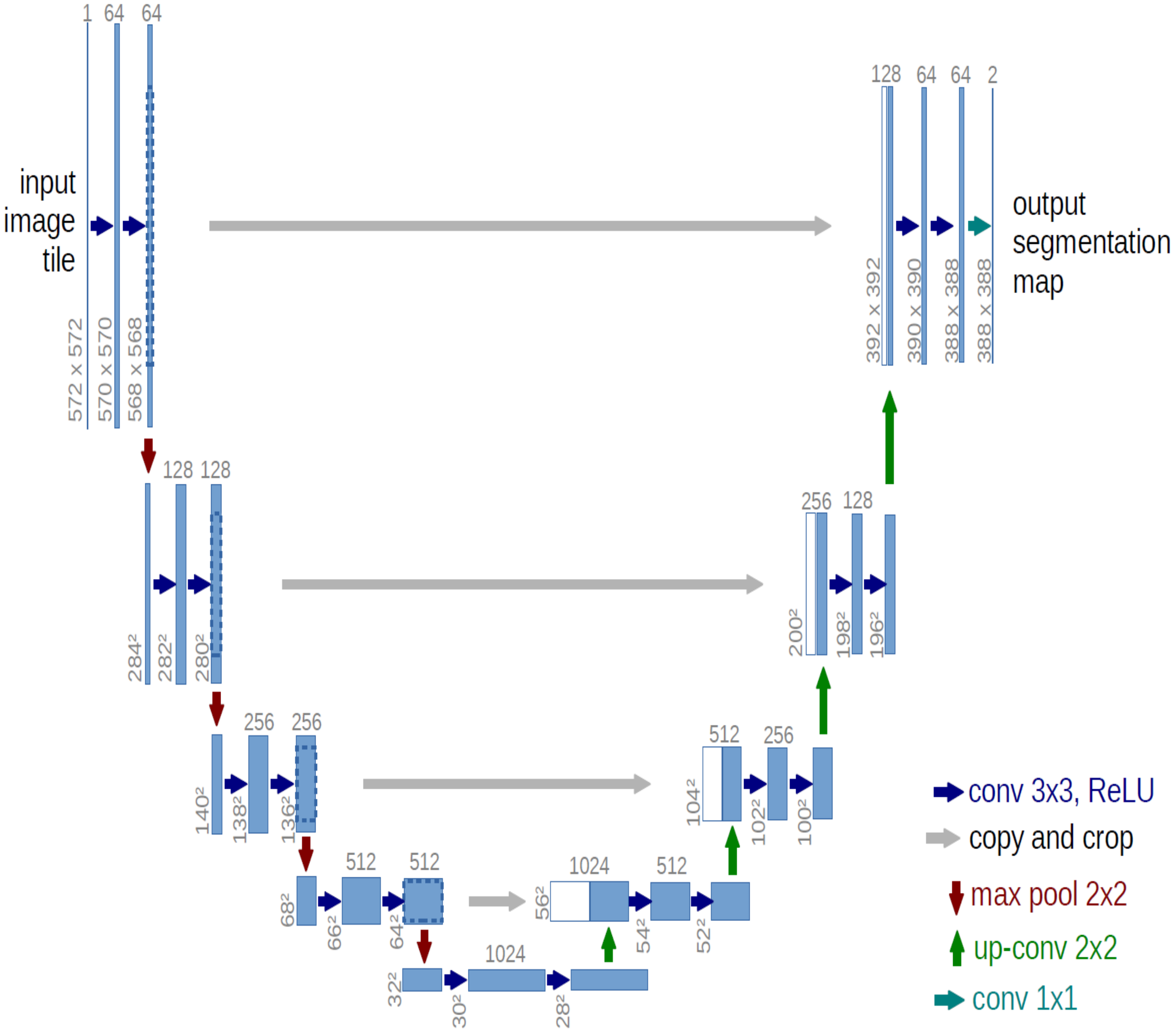
1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \mathbf{x}_{t-1} = \underline{\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\math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

$\mu_\theta(x_t, t)$ by eq. 11; see Slide 11

Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." *Advances in Neural Information Processing Systems* 33 (2020): 6840-6851.

Background: DDPM – Architecture

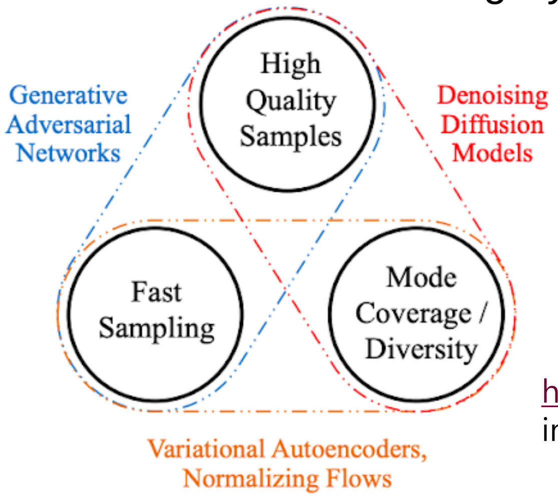
- Model: $\epsilon_\theta(x_t, t) = \hat{\epsilon}_{t-1}$
- Requirement: Same input and output dimensions
- UNet-like architecture commonly employed



https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/

Background: Diffusion vs. Other Generative Models

- vs. Normalising Flows: NF maps a data point to a latent variable following a deterministic trajectory (hence flow-based) making the mapping invertible, but the invertible map parametrised by NN may impose topological constraints compromising sampling quality. (Zhang, Qinsheng, and Yongxin Chen. "Diffusion normalizing flow." Advances in Neural Information Processing Systems 34 (2021): 16280-16291.)



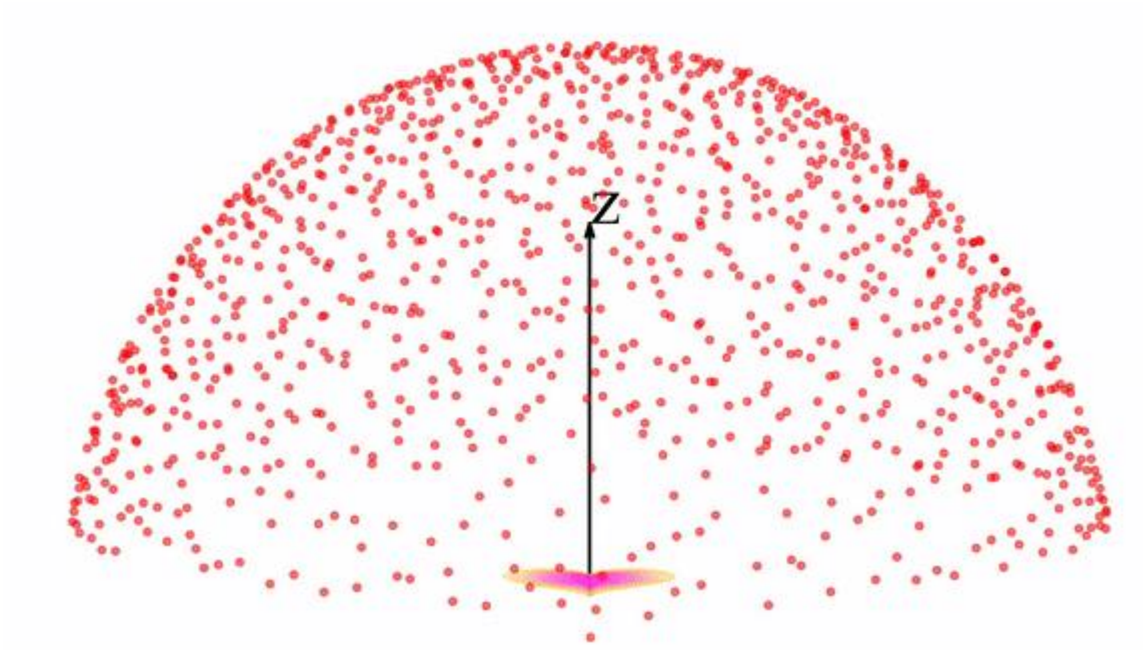
https://developer.nvidia.com/blog/improving-diffusion-models-as-an-alternative-to-gans-part-1/

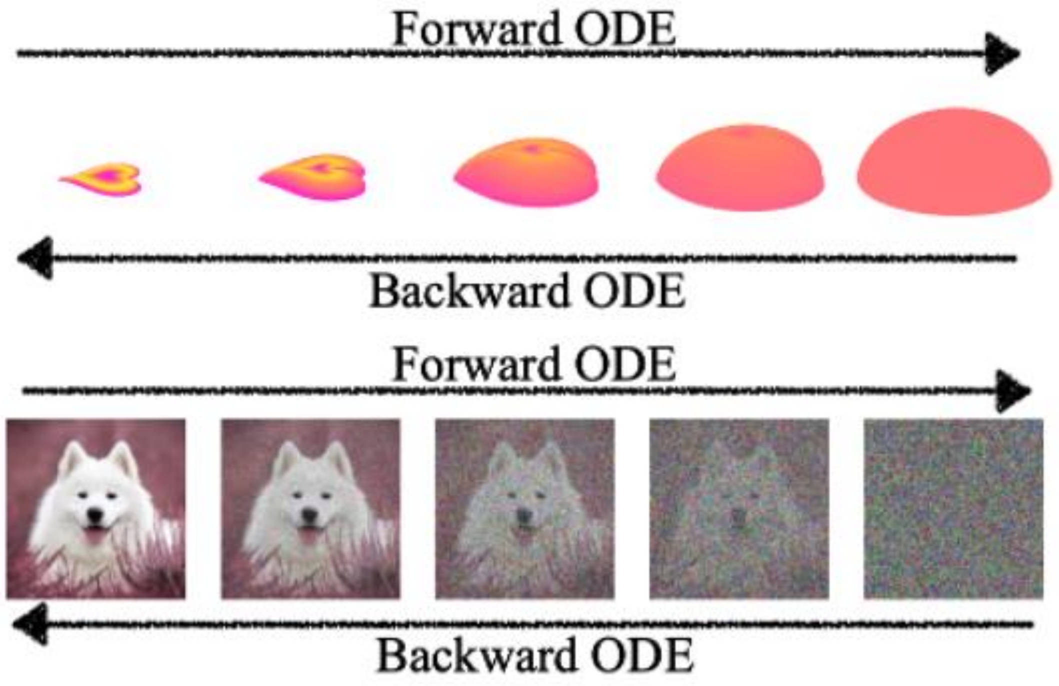
- Diffusion Model can suffer from having slow sampling, indirectly minimising VLB etc., which are topics of ongoing research (refer to Yang, Ling, et al. "Diffusion models: A comprehensive survey of methods and applications." arXiv preprint arXiv:2209.00796 (2022).)

Background: Diffusion vs. PFGM

- Poisson Flow Generative Models (PFGMs; NeurIPS 2022):
- Efficient flow-based “denoising” generator inspired by particle dynamics in an electric field.

Achieves “Diffusion Performance” while being 10-20 faster than the former on image generation tasks.

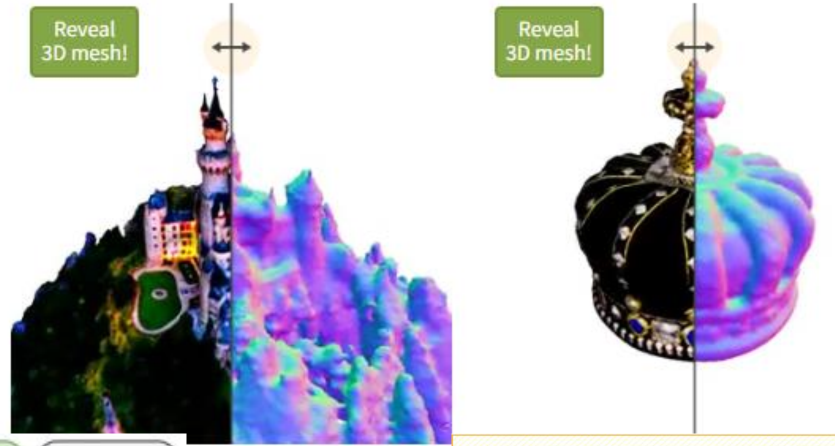


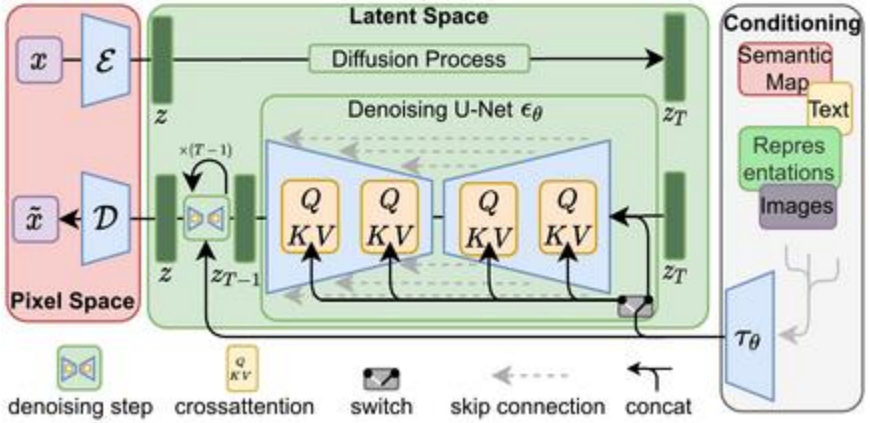


https://github.com/Newbeeer/Poisson_flow/blob/main/README.md

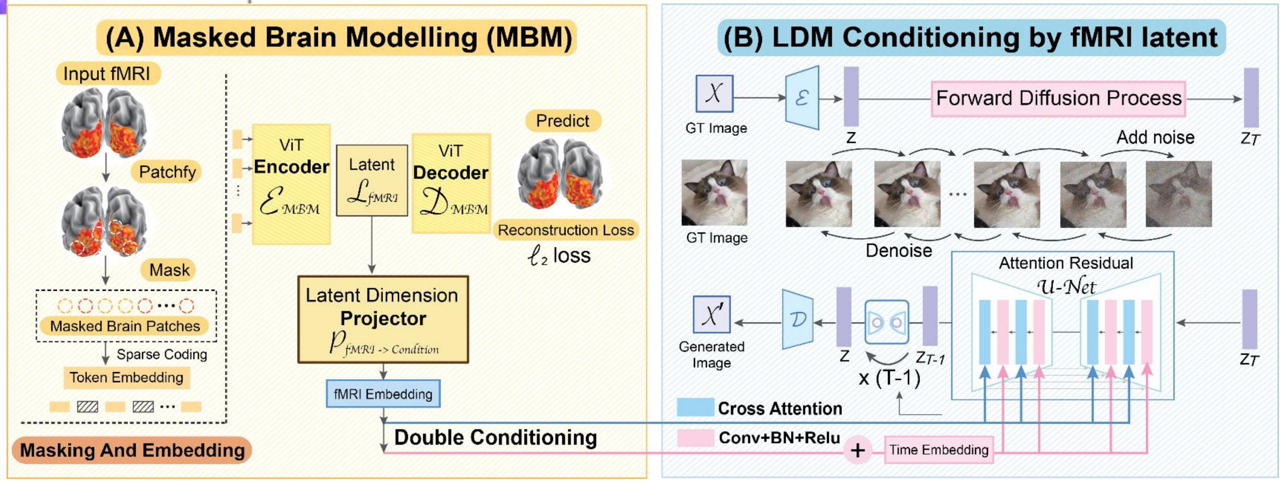
Background: Applications of Diffusion Models

- Image generation, 3D mesh generation, BCI, super-resolution etc.





stein Castle, aerial view.



Rombach, Robin, et al. "High-resolution image synthesis with latent diffusion models." *Proceedings of the IEEE/CVF Conference on CVPR*. 2022.
Lin, Chen-Hsuan, et al. "Magic3D: High-Resolution Text-to-3D Content Creation." *arXiv preprint arXiv:2211.10440* (2022).
Chen, Zijiao, et al. "Seeing Beyond the Brain: Conditional Diffusion Model with Sparse Masked Modeling for Vision Decoding." *arXiv preprint arXiv:2211.06956* (2022).

MotionDiffuse: Abstract

- Motion synthesis from text based on DDPM
  - Find-Grained control over whole body
  - Generate sequences of arbitrary length

https://youtu.be/U5PTnw490SA

MotionDiffuse: Model Summary

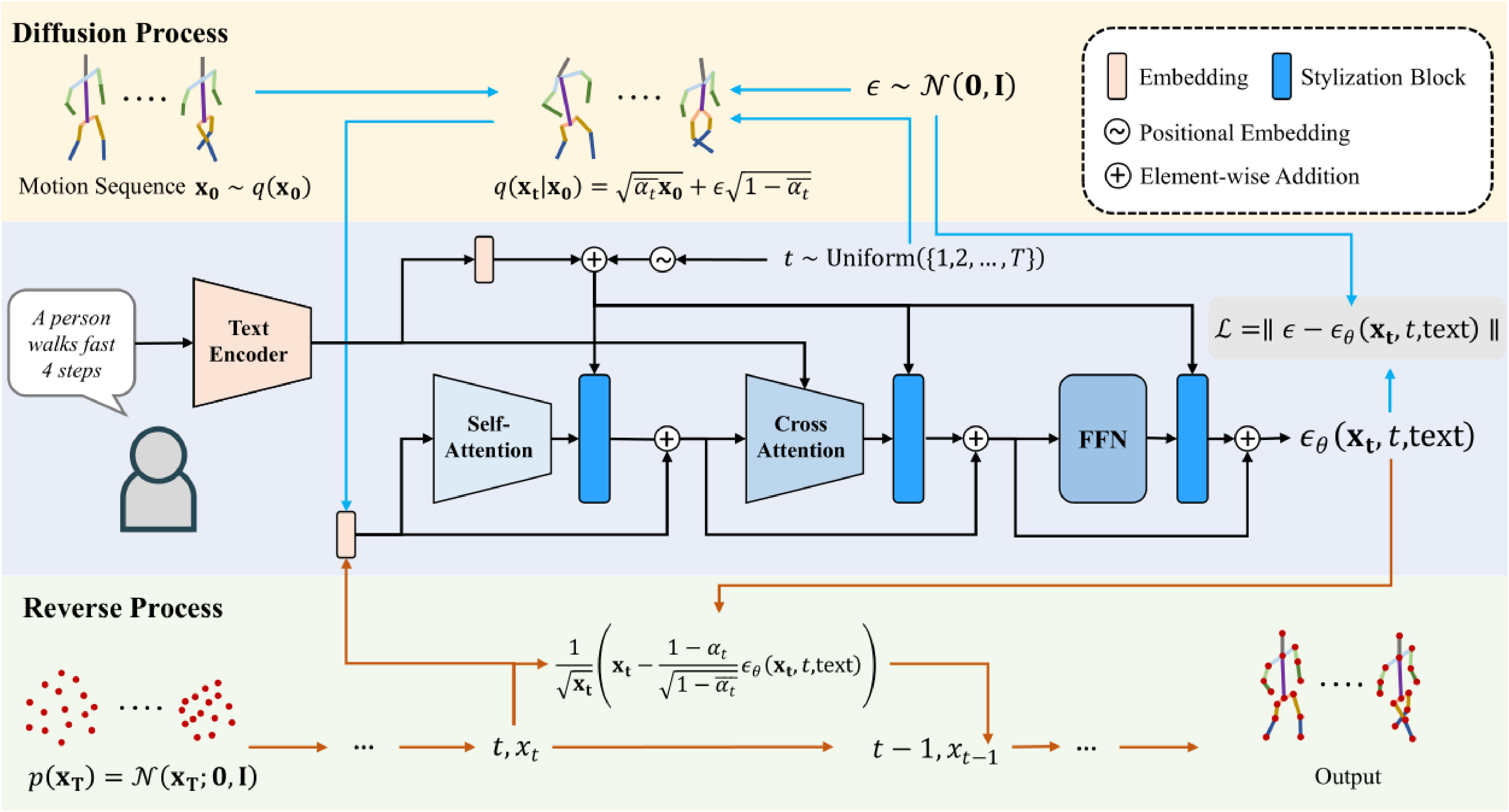


Fig. 2 Overall Pipeline of the proposed MotionDiffuse. The colors of the arrows indicate different stages: blue for training, red for inference, and black for both training and inference.

MotionDiffuse: Body Part-independent Controlling

- Complex prompts requiring independent actions from multiple body parts such as “running and waving left hand”, which may not be in the training d ataset, pose a challenge.
- Proposed method: *noise interpolation*, interpolate over noise calculated for each joint individually:

$$\overline{\epsilon}^{\text{part}} = \sum_{i=1}^{m} \epsilon_i^{\text{part}} \cdot M_i + \lambda_1 \cdot \nabla \Big(\sum_{1 \leq i,j \leq m} \| \epsilon_i^{\text{part}} - \epsilon_j^{\text{part}} \| \Big), \quad (10)$$

We have n text descriptions $\{text_i\}$ for different body parts, and we combine noise terms for each part

$$\epsilon_i^{\text{part}} = \epsilon_\theta(\mathbf{x}_t, t, \text{text}_i), \epsilon_i^{\text{part}} \in \mathbb{R}^{F \times D}$$

F: #frames, D: dimension of pose state (translation+rot)

$M_i \in \{0, 1\}^D$ is a binary vector to show which body part should we focus

MotionDiffuse: Results

Table 1 Quantitative results on the HumanML3D test set. All methods use the real motion length from the ground truth. '→' means results are better if the metric is closer to the real motions. We run all the evaluation 20 times and ± indicates the 95% confidence interval. The best results are in **bold**.

Methods	R Precision↑			FID↓	MultiModal Dist↓	Diversity→	MultiModality
	Top 1	Top 2	Top 3				
Real motions	$0.511^{\pm.003}$	$0.703^{\pm.003}$	$0.797^{\pm.002}$	$0.002^{\pm.000}$	$2.974^{\pm.008}$	$9.503^{\pm.065}$	-
Language2Pose	$0.246^{\pm.002}$	$0.387^{\pm.002}$	$0.486^{\pm.002}$	$11.02^{\pm.046}$	$5.296^{\pm.008}$	$7.676^{\pm.058}$	-
Text2Gesture	$0.165^{\pm.001}$	$0.267^{\pm.002}$	$0.345^{\pm.002}$	$7.664^{\pm.030}$	$6.030^{\pm.008}$	$6.409^{\pm.071}$	-
MoCoGAN	$0.037^{\pm.000}$	$0.072^{\pm.001}$	$0.106^{\pm.001}$	$94.41^{\pm.021}$	$9.643^{\pm.006}$	$0.462^{\pm.008}$	$0.019^{\pm.000}$
Dance2Music	$0.033^{\pm.000}$	$0.065^{\pm.001}$	$0.097^{\pm.001}$	$66.98^{\pm.016}$	$8.116^{\pm.006}$	$0.725^{\pm.011}$	$0.043^{\pm.001}$
Guo et al.	$0.457^{\pm.002}$	$0.639^{\pm.003}$	$0.740^{\pm.003}$	$1.067^{\pm.002}$	$3.340^{\pm.008}$	$9.188^{\pm.002}$	$2.090^{\pm.083}$
Ours	$\mathbf{0.491}^{\pm.001}$	$\mathbf{0.681}^{\pm.001}$	$\mathbf{0.782}^{\pm.001}$	$\mathbf{0.630}^{\pm.001}$	$\mathbf{3.113}^{\pm.001}$	$\mathbf{9.410}^{\pm.049}$	$1.553^{\pm.042}$

Appendix A: Proof of Slide 9 ★

Below is a derivation of Eq. (5), the reduced variance variational bound for diffusion models. This material is from Sohl-Dickstein et al. [53]; we include it here only for completeness.

$$L = \mathbb{E}_q\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] \tag{17}$$

$$= \mathbb{E}_q\left[-\log p(\mathbf{x}_T) - \sum_{t\geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}\right] \tag{18}$$

$$= \mathbb{E}_q\left[-\log p(\mathbf{x}_T) - \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} - \log \frac{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)}\right] \tag{19}$$

$$= \mathbb{E}_q\left[-\log p(\mathbf{x}_T) - \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)} \cdot \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)} - \log \frac{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)}\right] \tag{20}$$

$$= \mathbb{E}_q\left[-\log \frac{p(\mathbf{x}_T)}{q(\mathbf{x}_T|\mathbf{x}_0)} - \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)} - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\right] \tag{21}$$

$$= \mathbb{E}_q\left[D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T)) + \sum_{t>1} D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\right] \tag{22}$$

Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." *Advances in Neural Information Processing Systems* 33 (2020): 6840-6851.
and https://www.assemblyai.com/blog/diffusion-models-for-machine-learning-introduction - sup A