Diffusion model

Stéphane Nguyen

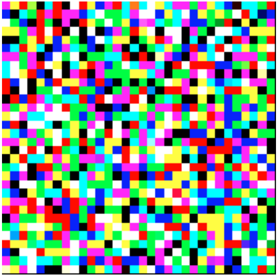
University of Geneva

stephane.nguyen@etu.unige.ch

December 19, 2023

Overview

Data from noise?



(a) Noise

(b) Generated horse

Diffusion models

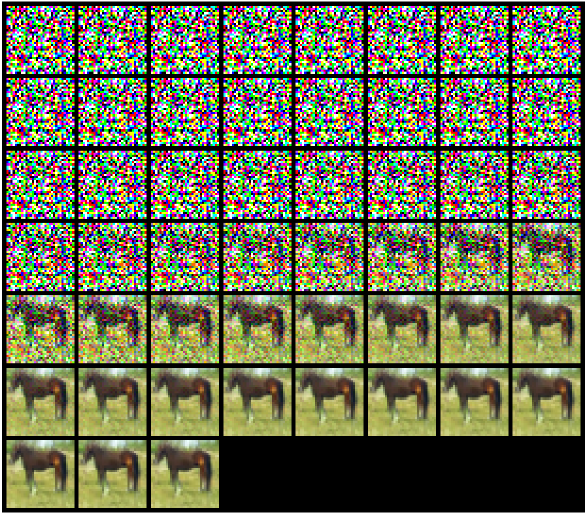


Figure: Iterative denoising process

Diffusion models

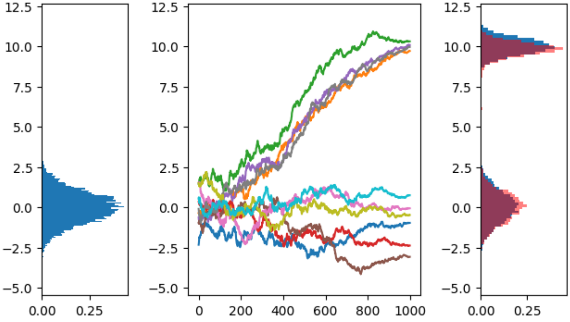


Figure: Pushing data towards high-density regions[1]

[1] With DDPM

Forward & reverse process

- Forward process: noise from data where p_{data} is unknown

$$x_i = y + \sigma_i \cdot z \sim p(x_i; \sigma_i), \quad y \sim p_{\text{data}}, z \sim \mathcal{N}(0, \text{Id})$$

Noise increases as time goes forward but i decreases.

Forward & reverse process

- Forward process: noise from data where p_{data} is unknown

$$x_i = y + \sigma_i \cdot z \sim p(x_i; \sigma_i), \quad y \sim p_{\text{data}}, z \sim \mathcal{N}(0, \text{Id})$$

Noise increases as time goes forward but i decreases.

- Reverse process: pushing data towards high-density regions using estimated noise level-dependent *score functions*

$$\nabla_x \log p(x; \sigma(t)), \quad \sigma(t_i) = \sigma_i$$

Noise decreases as time goes backward but i increases.

Probability Flow ODE

- Forward and reverse processes can be derived from continuous-time (stochastic) processes, solutions to forward and reverse-time (Stochastic) Differential Equations.

Probability Flow ODE

- Forward and reverse processes can be derived from continuous-time (stochastic) processes, solutions to forward and reverse-time (Stochastic) Differential Equations.
- We will only focus on the Probability Flow ODE [1, 2]

$$dx = -\dot{\sigma}(t)\sigma(t)\nabla_x \log p(x; \sigma(t))dt$$

where the schedule is set to $\sigma(t) = t$ to make "flow lines" more straight.

Probability Flow ODE

- Forward and reverse processes can be derived from continuous-time (stochastic) processes, solutions to forward and reverse-time (Stochastic) Differential Equations.
- We will only focus on the Probability Flow ODE [1, 2]

$$dx = -\dot{\sigma}(t)\sigma(t)\nabla_x \log p(x;\sigma(t))dt$$

  where the schedule is set to $\sigma(t) = t$ to make "flow lines" more straight.
- Deterministic and stochastic samplers of the EDM paper [1].
  - Numerical ODE solvers (+ explicit Langevin-like correction [1, 2])

Probability Flow ODE

- Forward and reverse processes can be derived from continuous-time (stochastic) processes, solutions to forward and reverse-time (Stochastic) Differential Equations.
- We will only focus on the Probability Flow ODE [1, 2]

$$dx = -\dot{\sigma}(t)\sigma(t)\nabla_x \log p(x; \sigma(t))dt$$

  where the schedule is set to $\sigma(t) = t$ to make "flow lines" more straight.
- Deterministic and stochastic samplers of the EDM paper [1].
  - Numerical ODE solvers (+ explicit Langevin-like correction [1, 2])
- Deterministic samplers generate different images for different initial conditions/noises.

Euler, Heun (actually RGK2)

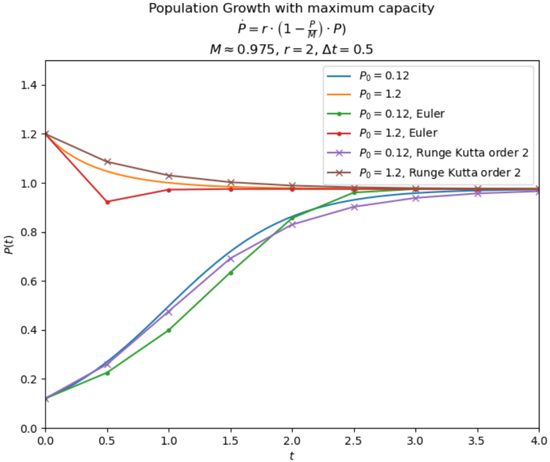


Figure: Euler and Heun (actually RGK2) on an toy example. Heun uses the average of two slopes: current and next.

Neural network preconditioning

- Don't directly train for $\nabla_x \log p(x; \sigma(t))$ or a denoiser $D_\theta(x; \sigma)$.

Neural network preconditioning

- Don't directly train for $\nabla_x \log p(x; \sigma(t))$ or a denoiser $D_\theta(x; \sigma)$.
- Neural network F_θ trained to predict the scaled [negative] noise or scaled original image from a noisy image.

Neural network preconditioning

- Don't directly train for $\nabla_x \log p(x; \sigma(t))$ or a denoiser $D_\theta(x; \sigma)$.
- Neural network F_θ trained to predict the scaled [negative] noise or scaled original image from a noisy image.
- Denoiser:

$$D_\theta(x; \sigma) = c_{\text{skip}}(\sigma)x + c_{\text{out}}(\sigma)F_\theta(c_{\text{in}}(\sigma)x; c_{\text{noise}}(\sigma))$$

- If $c_{\text{skip}} = 0$, F_θ has to predict the scaled original image.
- If $c_{\text{skip}} = 1$, F_θ has to predict the scaled [negative] noise.

$$c_{\text{skip}}(\sigma) = \frac{\sigma_{\text{data}}^2}{\sigma_{\text{data}}^2 + \sigma^2}$$

Neural network preconditioning

- Don't directly train for $\nabla_x \log p(x; \sigma(t))$ or a denoiser $D_\theta(x; \sigma)$.
- Neural network F_θ trained to predict the scaled [negative] noise or scaled original image from a noisy image.
- Denoiser:

$$D_\theta(x; \sigma) = c_{\text{skip}}(\sigma)x + c_{\text{out}}(\sigma)F_\theta(c_{\text{in}}(\sigma)x; c_{\text{noise}}(\sigma))$$

- If $c_{\text{skip}} = 0$, F_θ has to predict the scaled original image.
- If $c_{\text{skip}} = 1$, F_θ has to predict the scaled [negative] noise.

$$c_{\text{skip}}(\sigma) = \frac{\sigma_{\text{data}}^2}{\sigma_{\text{data}}^2 + \sigma^2}$$

- Don't want the identity function for noise-level extremes.

Neural network preconditioning

- Since

$$\nabla_x \log p(x; \sigma(t)) = \frac{D(x; \sigma) - x}{\sigma^2}$$

where $D(x; \sigma)$ is the optimal denoiser [1], we can go from F_θ to D_θ to the estimated $\nabla_x \log p(x; \sigma(t))$.

Neural network preconditioning

- Since

$$\nabla_x \log p(x;\sigma(t)) = \frac{D(x;\sigma) - x}{\sigma^2}$$

where $D(x;\sigma)$ is the optimal denoiser [1], we can go from F_θ to D_θ to the estimated $\nabla_x \log p(x;\sigma(t))$.

- Can get the slope to use in Euler or Heun methods!

$$\begin{aligned} \frac{dx}{dt} &= -t\nabla_x \log p(x;\sigma(t)) \\ \text{slope} &= -\frac{D_\theta(x;\sigma) - x}{\sigma} \end{aligned} \tag{1}$$

Neural network preconditioning

- Since

$$\nabla_x \log p(x;\sigma(t)) = \frac{D(x;\sigma) - x}{\sigma^2}$$

where $D(x;\sigma)$ is the optimal denoiser [1], we can go from F_θ to D_θ to the estimated $\nabla_x \log p(x;\sigma(t))$.

- Can get the slope to use in Euler or Heun methods!

$$\begin{aligned} \frac{dx}{dt} &= -t\nabla_x \log p(x;\sigma(t)) \\ \text{slope} &= -\frac{D_\theta(x;\sigma) - x}{\sigma} \end{aligned} \tag{1}$$

- Need for dense prediction $\implies$ U-Net architecture

Original U-Net for dense prediction [3]

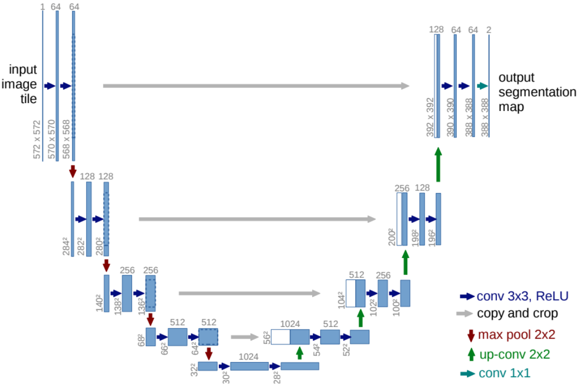


Figure: Original U-Net architecture taken from their paper [3]

Except if otherwise mentionned, our models are U-Net's with self-attention mechanisms (mostly in low resolution layers) and conditional batch-norm. Down: avg pool, Up: nearest-interp. & 3×3 conv.

Models

A total of 10 models

- Unconditional generation:
  - FashionMNIST, CIFAR-10: 1 model each
  - CelebA: 4 models: tiny no self attention, tiny, small, big

Models

A total of 10 models

- Unconditional generation:
  - FashionMNIST, CIFAR-10: 1 model each
  - CelebA: 4 models: tiny no self attention, tiny, small, big
- Conditional generation with Classifier-Free Guidance (CFG)
  - FashionMNIST, CIFAR-10: 2 models each: one with and the other without self-attention

Models

A total of 10 models

- Unconditional generation:
  - FashionMNIST, CIFAR-10: 1 model each
  - CelebA: 4 models: tiny no self attention, tiny, small, big
- Conditional generation with Classifier-Free Guidance (CFG)
  - FashionMNIST, CIFAR-10: 2 models each: one with and the other without self-attention
- Class-conditional models are sometimes trained with a special id to learn the unconditional score function.

Models

A total of 10 models

- Unconditional generation:
  - FashionMNIST, CIFAR-10: 1 model each
  - CelebA: 4 models: tiny no self attention, tiny, small, big
- Conditional generation with Classifier-Free Guidance (CFG)
  - FashionMNIST, CIFAR-10: 2 models each: one with and the other without self-attention
- Class-conditional models are sometimes trained with a special id to learn the unconditional score function.
- CFG sampling is done by using a weighted sum of the unconditional and conditional score functions.

Results

Fréchet Inception Distance (FID). FashionMNIST

Class-conditional FID, cfg.scale 1			
	Train*	Val*	Test
w/ self-attention	9.1676	10.5069	**10.2580**
w/o self-attention	13.9130	15.0970	14.7425
Unconditional FID			
	Train*	Val*	Test
w/ self-attention	19.1392	20.1931	19.5964

Fréchet Inception Distance (FID). CIFAR-10

Class-conditional FID, cfg.scale 2.5			
	Train*	Val*	Test
w/ self-attention	20.4102	22.4736	**22.5495**
w/o self-attention	22.7896	24.7956	24.3233
Unconditional FID			
	Train*	Val*	Test
w/ self-attention	28.5462	30.6864	30.6589

Fréchet Inception Distance (FID). CelebA

Unconditional "tiny" model (30-35M params.)			
	Train*	Val*	Test
w/ self-attention	22.0655	23.1863	**21.8915**
w/o self-attention	26.6725	27.6983	26.3588
Unconditional "small" model (80M params.)			
	Train*	Val*	Test
w/ self-attention	17.8853	18.9739	18.2885

Notes:

- w/ self-attention: not at each resolution level like FashionMNIST and CIFAR-10.
- Unconditional FID of the "big" model (81 M parameters) is not reported here.

Results with self-attention

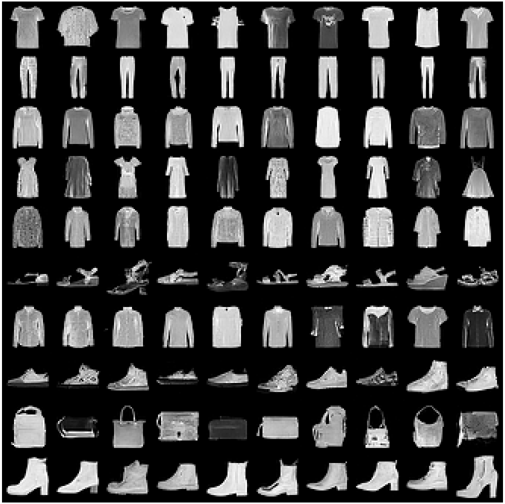


Figure: FashionMNIST cfg.scale=1, 100 epochs, 50 Euler method steps

Results with self-attention

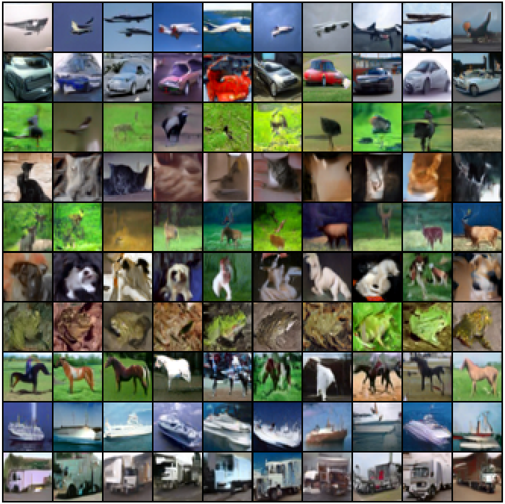


Figure: CIFAR-10 cfg.scale=2.5, 200 epochs, 50 Euler method steps

Results CelebA tiny

Figure: CelebA 35M parameters, 125 epochs, 50 stochastic Heun method steps

Other results

- https://github.com/Zenchiyu/diffusion
- https://github.com/Zenchiyu/diffusion/results

Demo

The End

A few design choices from the EDM paper [1]

- Sampling
  - Schedule $\sigma(t) = t$ and scaling $s(t) = 1$ to make "flow lines" more straight.
  - More (time-)steps[2] when less noise, i.e. regions where Euler method would suffer

$$\left(\sigma_{\max}^{1/\rho} + \frac{i}{N-1}\left(\sigma_{\min}^{1/\rho} - \sigma_{\max}^{1/\rho}\right)\right)^{\rho}$$

- Training
  - Specific noise distribution (log-normal) to focus training effort on some noise levels.
- Network preconditioning
  - Ensure that the network is never asked to perform a trivial task
  - Control the magnitude that would immensely vary depending on noise level
  - Predict the noise or predict the image?
  - Re-scaling the input?

[2] $\neq \sigma(t)$

Output of the denoiser

- Slope directly points to the output of the denoiser for $\sigma(t) = t, s(t) = 1$.

$$
\begin{aligned}
\frac{dx}{dt} &= -t\nabla_x \log p(x; \sigma(t)) \\
\text{slope} &= -\frac{D_\theta(x;\sigma) - x}{t} \\
D_\theta(x;\sigma) &= x - t \cdot \text{slope}
\end{aligned}
\tag{2}
$$

References

Figures without references come from author of the slides.

[1] Karras, T., Aittala, M., Aila, T., and Laine, S. (2022)
Elucidating the Design Space of Diffusion-Based Generative Models.
arxiv:2206.00364 [cs, stat]

[2] Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2021)
Score-Based Generative Modeling through Stochastic Differential Equations.
arXiv:2011.13456 [cs, stat]

[2] Ronneberger, O., Fischer, P., and Brox, T. (2015)
U-Net: Convolutional Networks for Biomedical Image Segmentation.
arXiv:1505.04597 [cs]