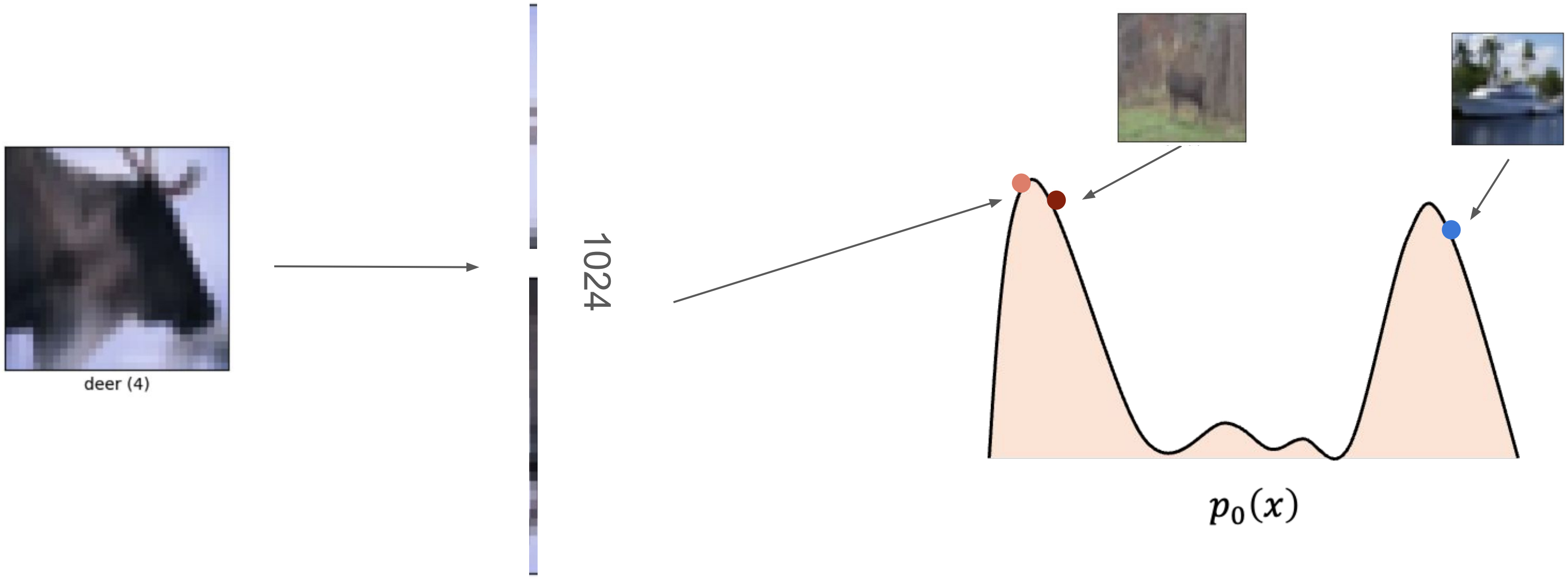
Privacy Issues in DGMs: How to detect & mitigate

Dongjae Jeon

Paper: Icml link

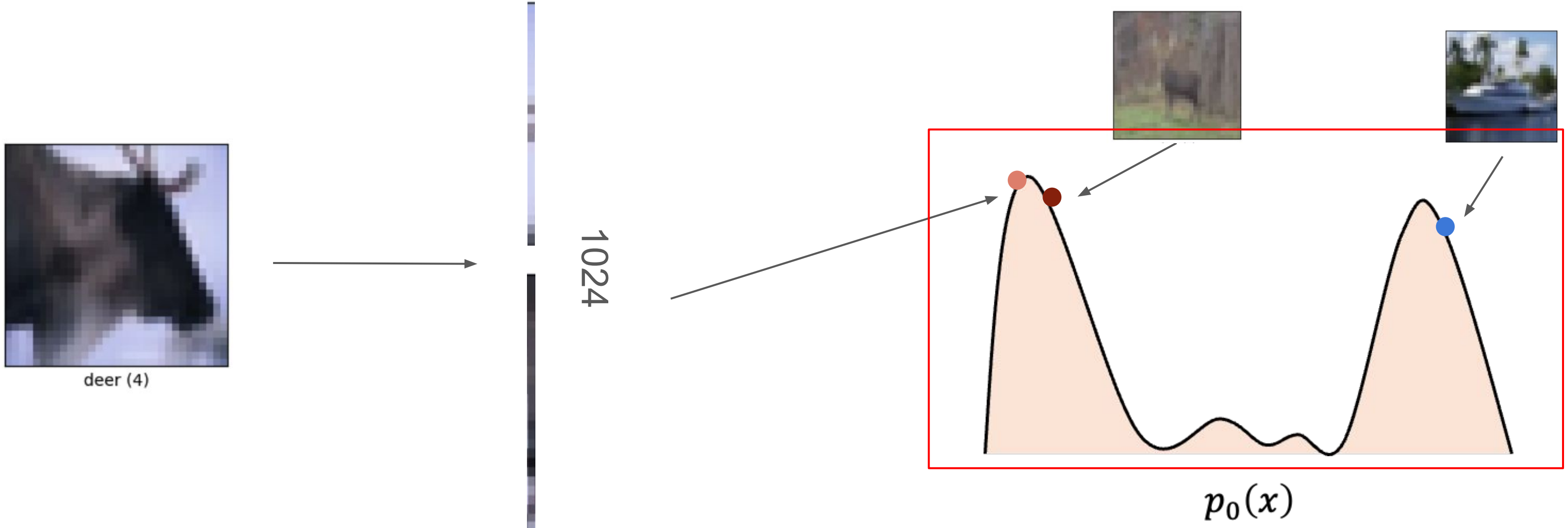
1. Detailed Background on Diffusion Model

Image = 1024 sized vector = lives in 1024 **p(x)** (we don't know)

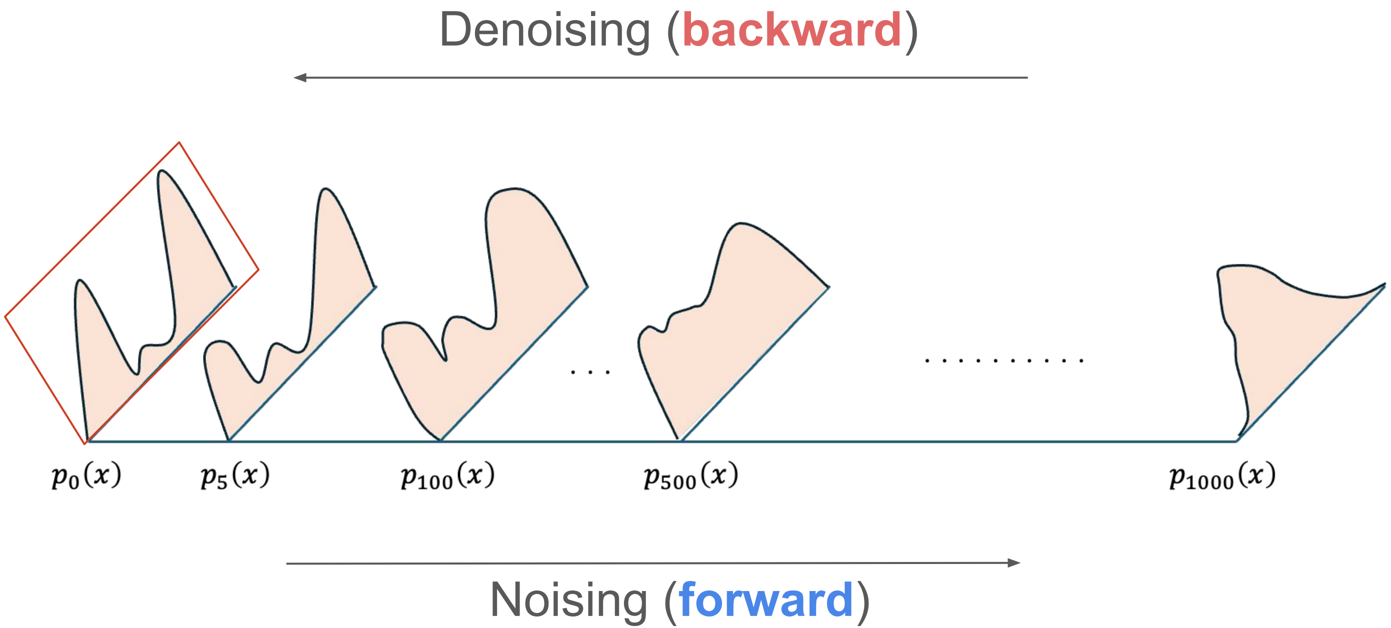


1. Detailed Background on Diffusion Model

Image = 1024 sized vector = lives in 1024 **p(x)** (we don't know)



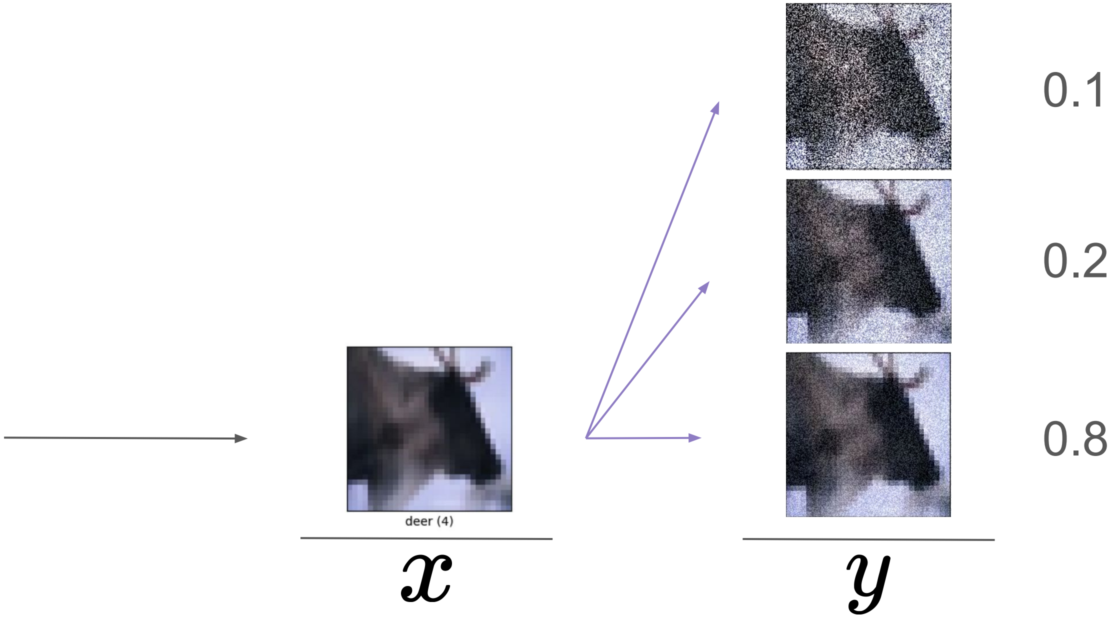
1. Detailed Background on Diffusion Model (cont'd)



1. Detailed Background on Diffusion Model (cont'd)

$x =$ 

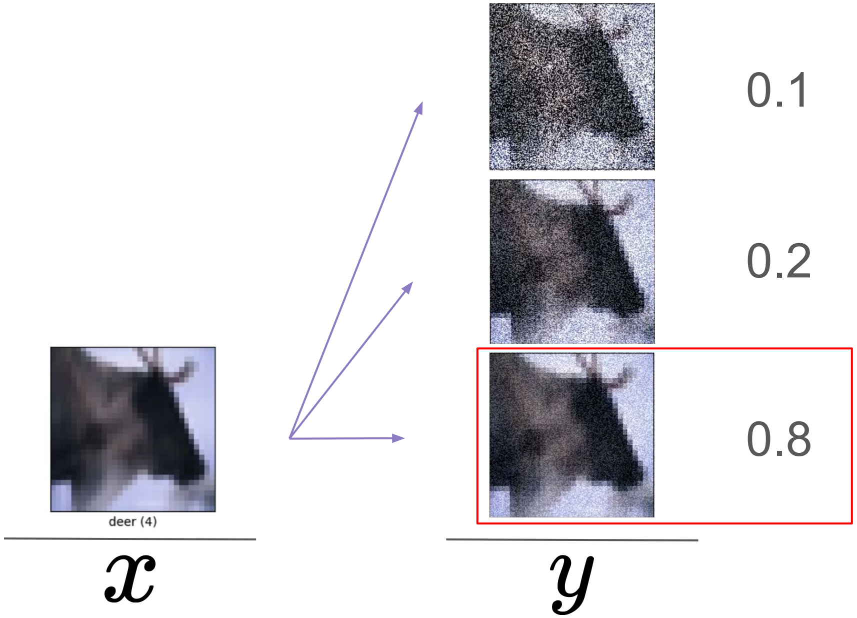
$$p(y|x)$$



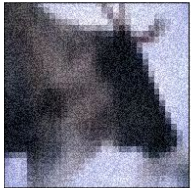
1. Detailed Background on Diffusion Model (cont'd)

$x =$ 

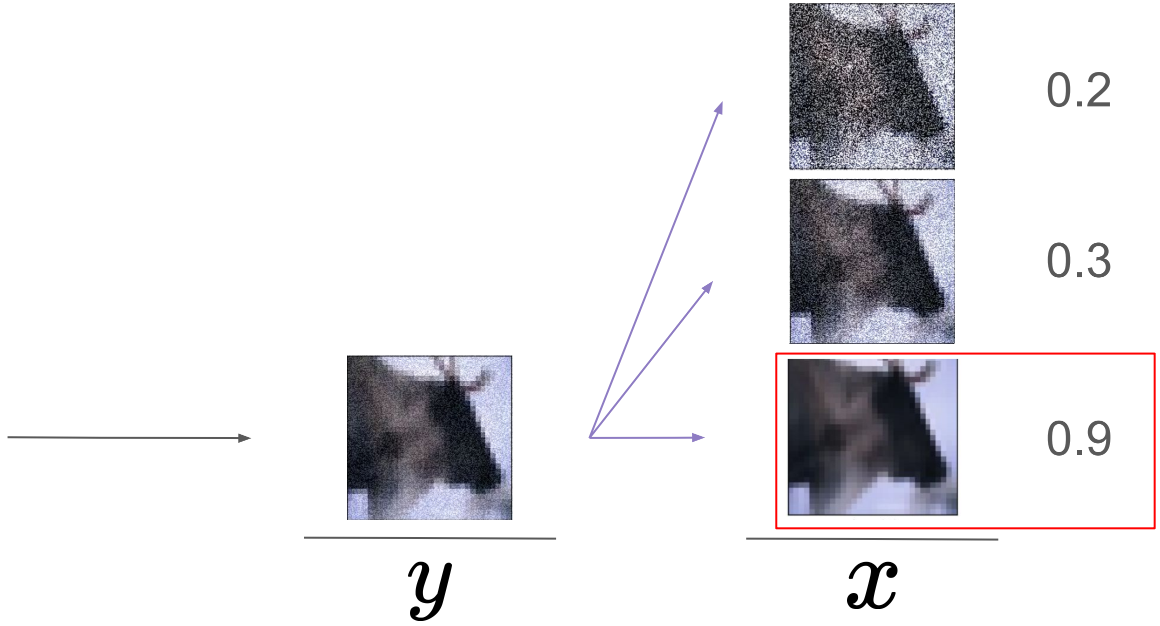
$$p(y|x)$$



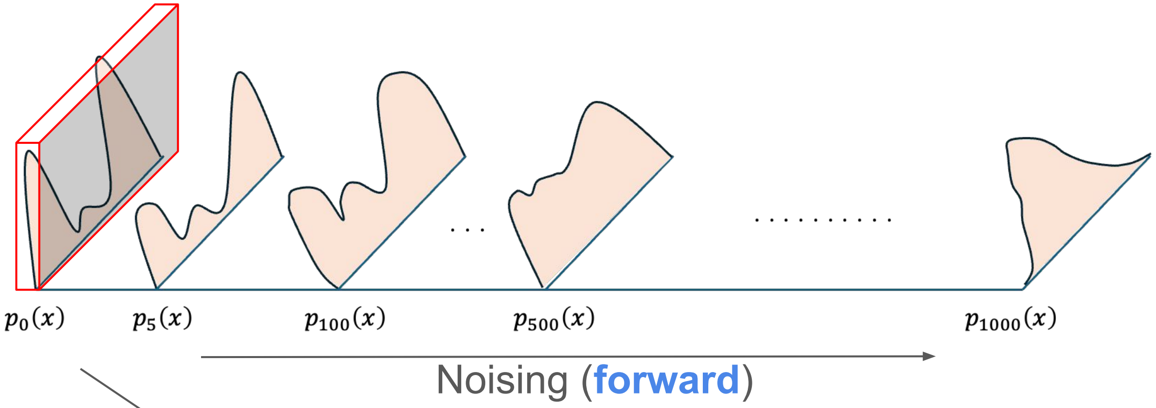
1. Detailed Background on Diffusion Model (cont'd)

$y =$ 

$p(x|y)$



1. Detailed Background on Diffusion Model (cont'd)



$$y = x + z, \;\; z \sim \mathcal{N}(0, I)$$

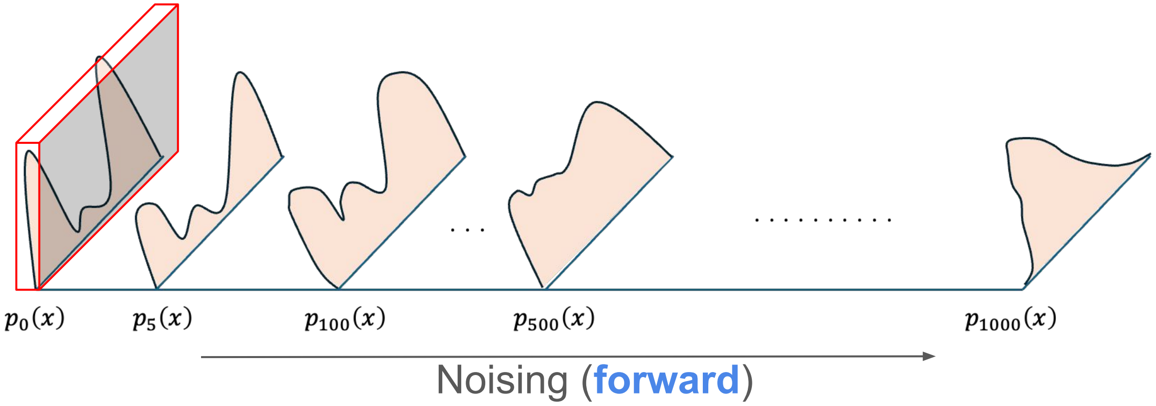
$$y \sim \mathcal{N}(x, I)$$

$$p_{1|0}(y|x) = \mathcal{N}(y; x, I)$$

Do this in every time t $\quad p_{t|t-1}(x_t \mid x_{t-1}) = \mathcal{N}(x_t;\; x_{t-1}, I)$

Gaussian convolution demo: https://phiresky.github.io/convolution-demo/

1. Detailed Background on Diffusion Model (cont'd)



$p_{1000}(x) \neq N(0, I)$

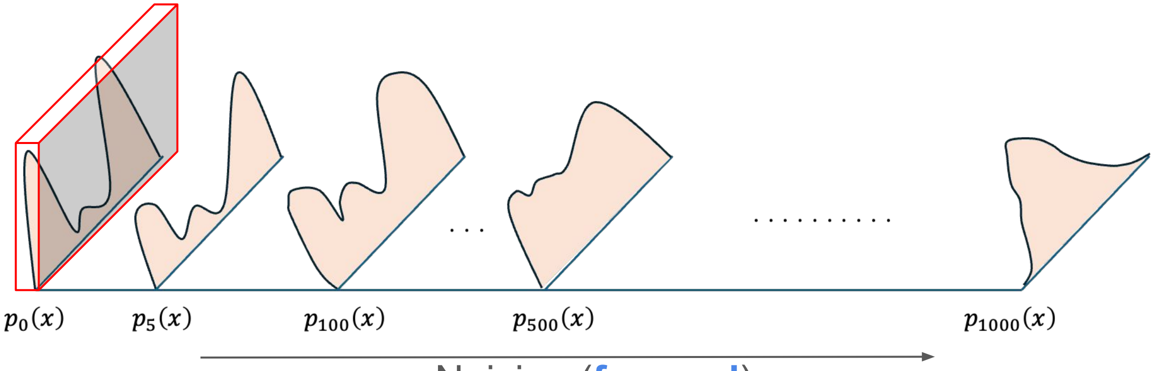
Then, how to start sampling?

$$p_{t|t-1}(x_t \mid x_{t-1}) = \mathcal{N}(x_t;\ x_{t-1}, I)$$ Variance Exploding

$$p_{t|t-1}(x_t \mid x_{t-1}) = \mathcal{N}\left(x_t;\ \sqrt{1-\beta_t}\, x_{t-1},\ \beta_t I\right)$$ Variance Preserving

Suppress to prevent exploding variance.

1. Detailed Background on Diffusion Model (cont'd)



Noising (**forward**)

$p_{1000}(x) \neq N(0, I)$

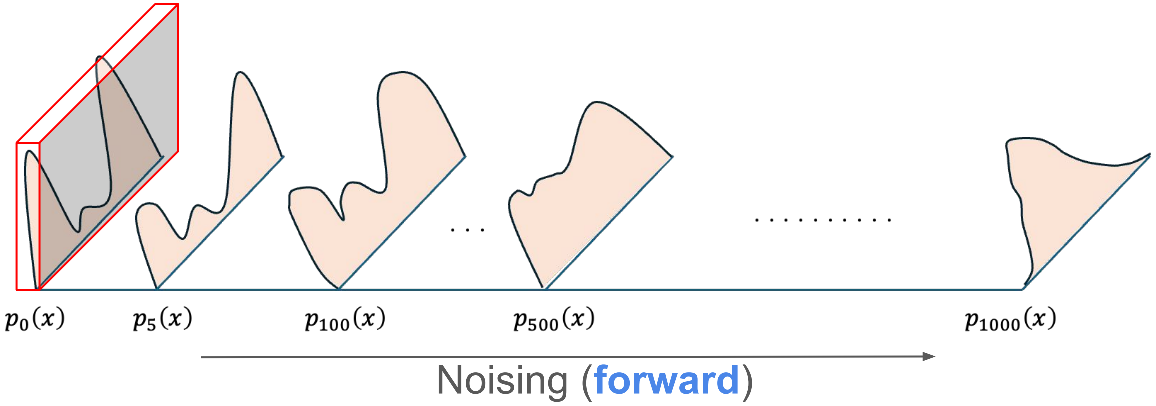
Then, how to start sampling?

$$p_{t|t-1}(x_t \mid x_{t-1}) = \mathcal{N}(x_t;\ x_{t-1}, I)$$ Variance Exploding

$$p_{t|t-1}(x_t \mid x_{t-1}) = \mathcal{N}\left(x_t;\ \sqrt{1-\beta_t}\, x_{t-1},\ \beta_t I\right)$$ Variance Preserving

Suppress to prevent exploding variance.

1. Detailed Background on Diffusion Model (cont'd)

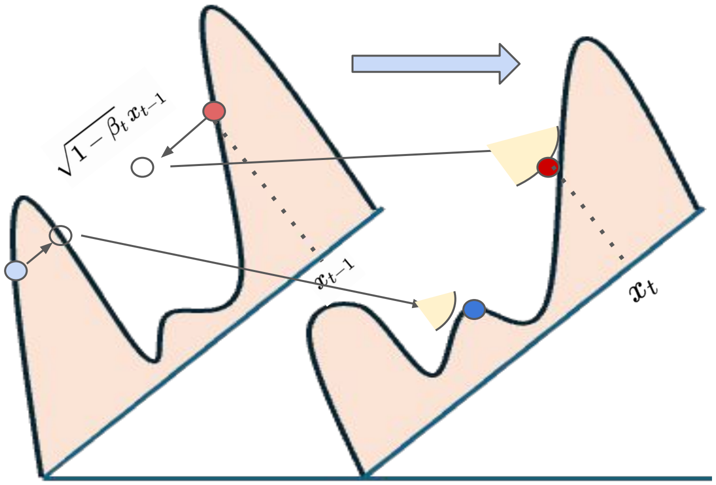


$$p_{t|t-1}(x_t \mid x_{t-1}) = \mathcal{N}(x_t;\ x_{t-1}, I)$$ Variance Exploding

$$p_{t|t-1}(x_t \mid x_{t-1}) = \mathcal{N}\left(x_t;\ \sqrt{1-\beta_t}\, x_{t-1},\ \beta_t I\right)$$ Variance Preserving

Our interest

1. Detailed Background on Diffusion Model (cont'd)



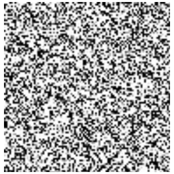
$$p_{t|t-1}(x_t \mid x_{t-1}) = \mathcal{N}\left(x_t;\, \sqrt{1-\beta_t}\, x_{t-1},\, \beta_t I\right)$$

$$p_{1000}(x) = \mathcal{N}(0, I)$$ No matter what p0(x) is.

1. Detailed Background on Diffusion Model (cont'd)

$$p_{1000}(x) = \mathcal{N}(0, I)$$

$$\text{Sample} \quad k \sim \mathcal{N}(0, I)$$ (What we can sample.)

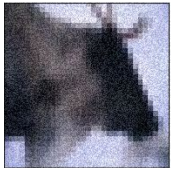


for 500 x

$$p_{t-1|t}(x_{t-1}|x_t)$$

for 500 x

$$p_{t-1|t}(x_{t-1}|x_t)$$



1. Detailed Background on Diffusion Model (cont'd)

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

$$p_{t-1|t}(x_{t-1}|x_t) \approx \mathcal{N}\Big(x_{t-1};\ BLANK,\ \tilde{\beta}_t I\Big)$$

Tweedie's formula

$$BLANK = \frac{1}{\sqrt{1-\beta_t}}\left(x_t + \beta_n \frac{\partial}{\partial x_t} \log p_t(x_t)\right)$$

tweedie: https://efron.ckirby.su.domains/papers/2011TweediesFormula.pdf

1. Detailed Background on Diffusion Model (cont'd)

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

$$p_{t-1|t}(x_{t-1}|x_t) \approx \mathcal{N}\left(x_{t-1};\ BLANK,\ \tilde{\beta}_t I\right)$$

Tweedie's formula

Neural Network

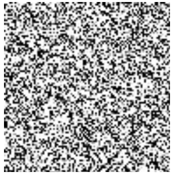
$$BLANK = \frac{1}{\sqrt{1-\beta_t}}\left(x_t + \beta_n \frac{\partial}{\partial x_t} \log p_t(x_t)\right) \rightarrow s_t(x_t, t)$$

also called "Score"

1. Detailed Background on Diffusion Model (cont'd)

$$p_{1000}(x) = \mathcal{N}(0, I)$$

$$\text{Sample} \quad k \sim \mathcal{N}(0, I)$$ (What we can easily sample)

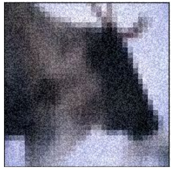


for 500 x

$$s_t(x_t, t)$$

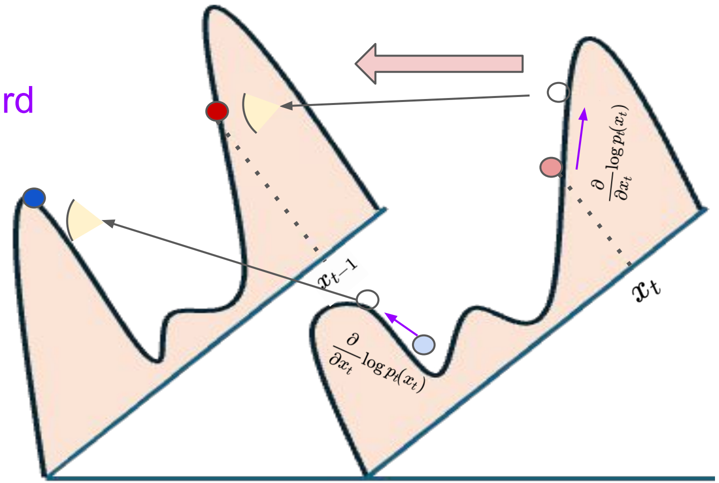
for 500 x

$$s_t(x_t, t)$$



1. Detailed Background on Diffusion Model (cont'd)

Gradually moving samples toward **higher likelihood region** using **"gradient"** information.



$$p_{t-1|t}(x_{t-1}|x_t) \approx \mathcal{N}\Big(x_{t-1};\, BLANK,\, \tilde{\beta}_t I\Big)$$

Tweedie's formula

$$BLANK = \frac{1}{\sqrt{1-\beta_t}}\left(x_t + \beta_n \frac{\partial}{\partial x_t} \log p_t(x_t)\right)$$

Neural Network

$$s_t(x_t, t)$$

1. Detailed Background on Diffusion Model (cont'd)

Takeaway:

1) Diffusion Models learn gradient of $p_t(x)$: $\frac{\partial}{\partial x_t} \log p_t(x_t)$

2) We can not sample from data distribution directly, but, we can sample from **Gaussian**, and **gradually pushing** it as a "real-like" image.

2. Memorization in Diffusion Models

Exact mem.

Training Image | Generated Image

“Living in the Light with Ann Graham Lotz”

Partial mem.

Training Image | Generated Image

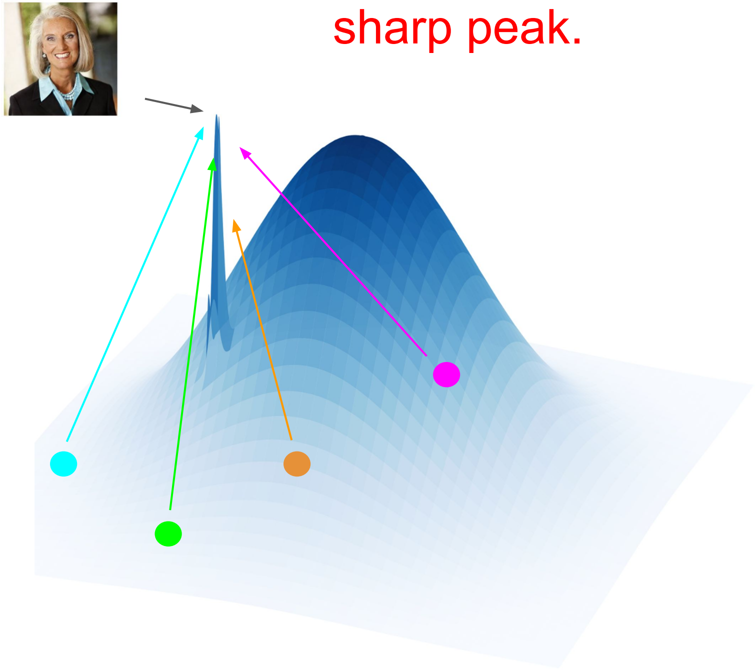
“Plattville Green Area Rug by Andover Mills”

Image credit: https://arxiv.org/pdf/2407.21720

3. What does it mean to be memorized?

Generated Image

Data point lies on a sharp peak.

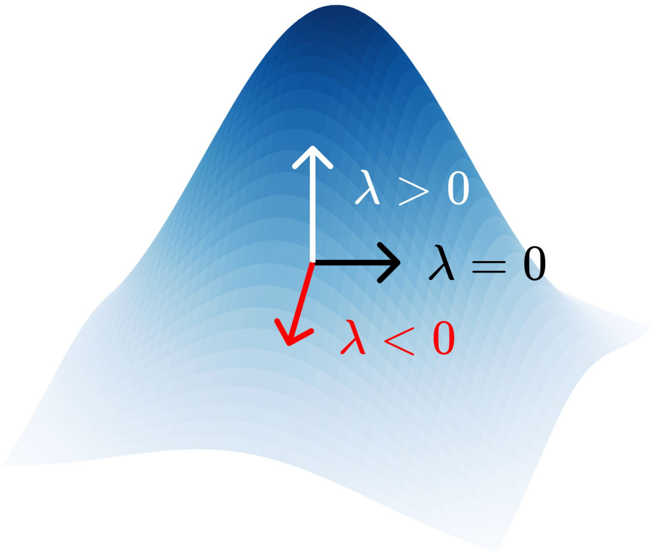


$p_0^\theta(x)$

4. How can we detect it?

$$s_t(x_t, t) = \frac{\partial}{\partial x_t} \log p_t(x_t)$$

$$\frac{\partial}{\partial x_t} s_t(x_t, t) = \frac{\partial^2}{\partial x_t^2} \log p_t(x_t)$$



Hessian Eigenvalues tell Curvature:

- $\lambda \geq 0$: Concave downward or Flat
- $\lambda < 0$: Concave upward (Key for finding peaks)

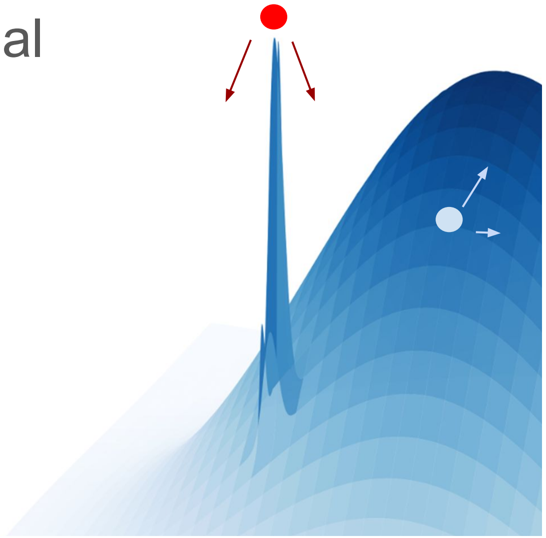
4. How can we detect it? (cont'd)

Hessian Eigenvalues tell Curvature:

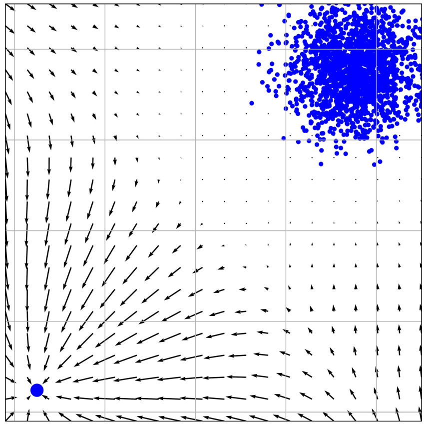
- $\lambda \geq 0$: Concave downward or Flat
- $\lambda < 0$: Concave upward (Key for finding peaks)

Memorized sample should reveal
large negative eigenvalues,

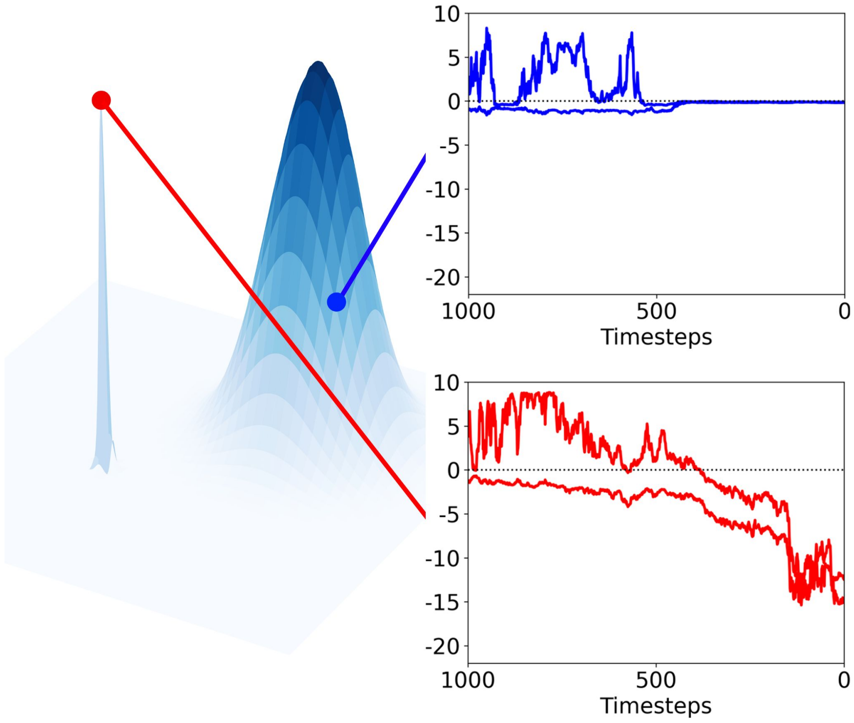
while non-memorized show
positive eigenvalues



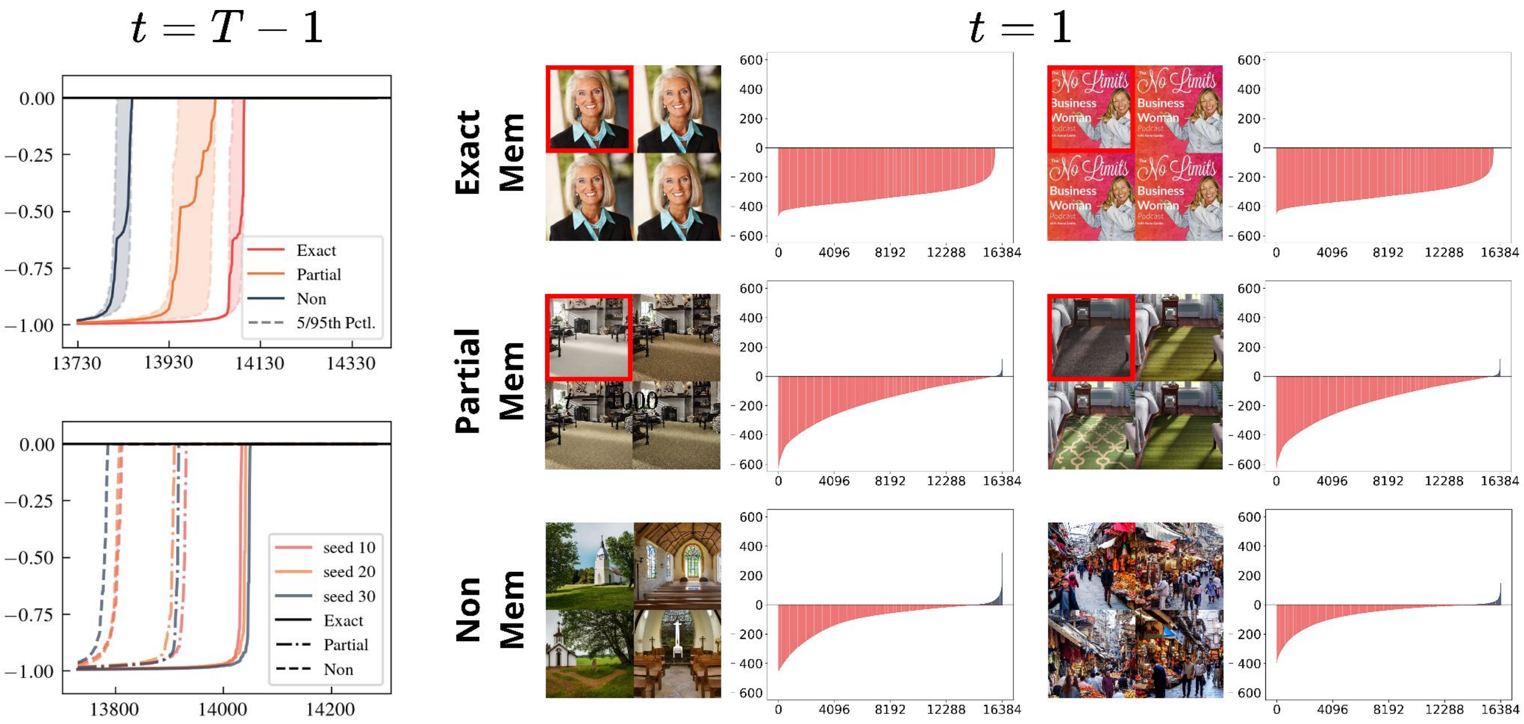
4. How can we detect it? (cont'd)



Train data



4. How can we detect it? (cont'd)



Eigenvalues in Stable Diffusion

4. How can we detect it? (cont'd)

But, doing **backpropagation** in Stable Diffusion is nonsense
We use the sum of eigenvalues as a proxy!

Very cheap to compute.

$$\mathbb{E}\left[\|s_t(x_t,t)\|^2\right] = -\operatorname{Tr}\left(H_t(x_t,t)\right) = -\sum_{i=1}^{d}\lambda_i,$$

Under gaussian assumption,

$$\mathbb{E}\left[\|H_t(x_t,t)\, s_t(x_t,t)\|^2\right] = -\operatorname{Tr}\left(H_t(x_t,t)^3\right) = -\sum_{i=1}^{d}\lambda_i^3.$$

4. How can we detect it? (cont'd)

Method	Steps	n	SD v1.4 AUC	SD v1.4 TPR@1%FPR	SD v2.0 AUC	SD v2.0 TPR@1%FPR
Tiled ℓ_2 (Carlini et al., 2023)	50	4	0.908	0.088	0.792	0.114
		16	0.94	0.232	0.907	0.114
LE (Ren et al., 2024)	1	1	0.846	0.116	0.848	0
		4	0.839	0.13	0.853	0
		16	0.832	0.124	0.851	0
AE (Ren et al., 2024)	50	1	0.606	0	0.809	0
		4	0.628	0	0.82	0
		16	0.598	0	0.817	0
BE (Chen et al., 2024)	50	1	0.986	0.95	0.983	0.908
		4	0.997	0.98	0.99	0.945
		16	0.997	0.982	0.99	0.949
$\|s_\theta^\Delta(\mathbf{x}_t)\|$ (Wen et al., 2024)	1	1	0.976	0.896	0.948	0.739
		4	0.992	0.944	0.98	0.876
		16	0.99	0.928	0.983	0.881
	5	1	0.991	0.932	0.969	0.885
		4	0.997	0.978	0.984	0.917
		16	**0.998**	**0.982**	0.987	0.931
	50	1	0.983	0.948	0.982	0.904
		4	0.996	**0.982**	0.99	**0.949**
		16	**0.998**	0.98	**0.991**	0.945
$\|H_\theta^\Delta(\mathbf{x}_T)s_\theta^\Delta(\mathbf{x}_T)\|^2$ (Ours)	1	1	0.987	0.908	0.959	0.74
		4	**0.998**	**0.982**	**0.991**	0.895

5. How can we mitigate it?

Previous approaches,

[1] Change text prompts
[2] Put random tokens between prompts
[3] Weaken text-conditioning during sampling
…..

Degrade user utility and image quality!!

5. How can we mitigate it? (cont'd)

5. How can we mitigate it? (cont'd)

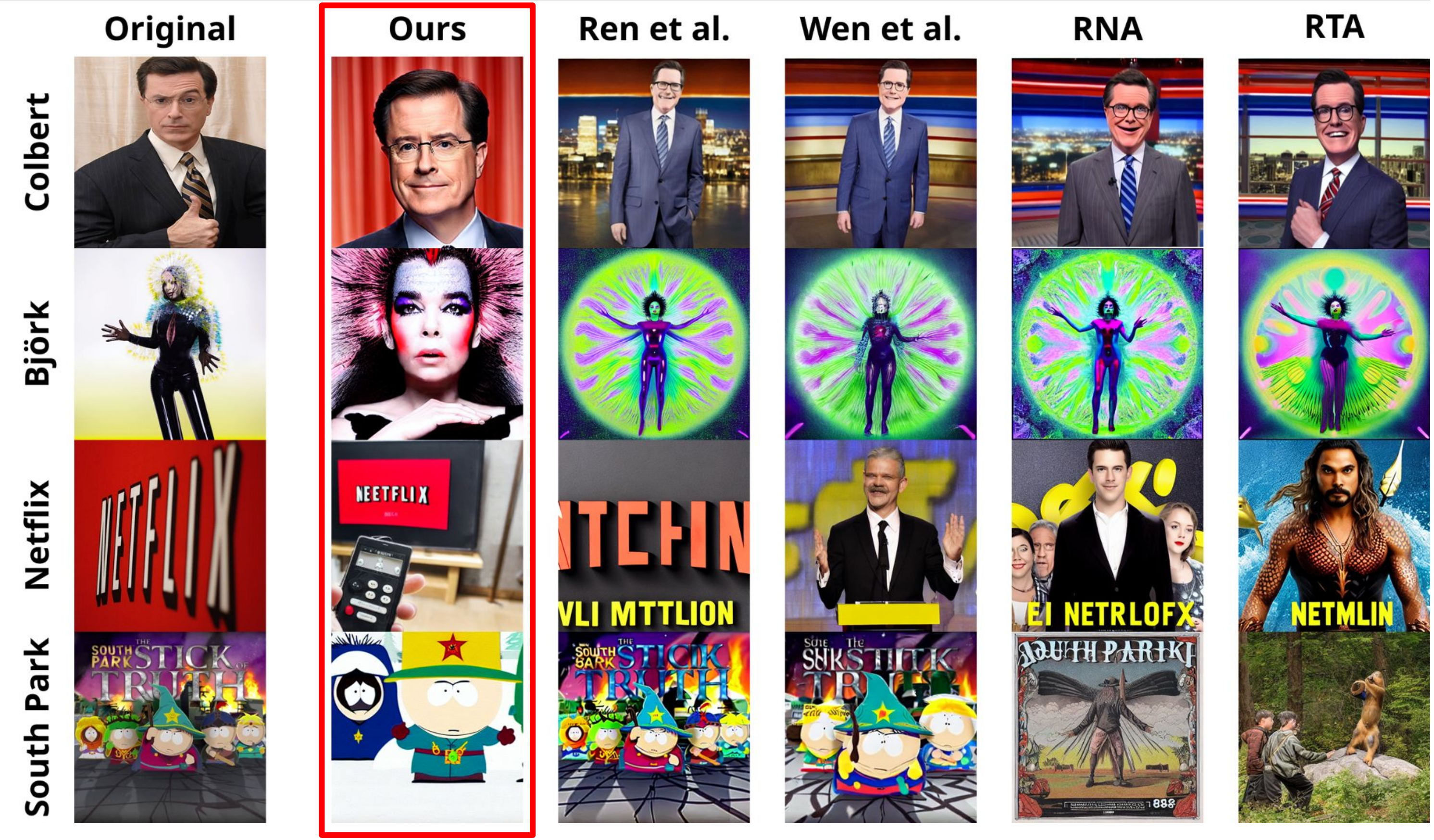
ODE samplers have **1 to 1** relationship between (Xt, Image)
Memorization is revealed even at the first timestep!

Why don't we just start sampling from
Gaussian latent on less sharper landscape?
(a.k.a Seed sampling)

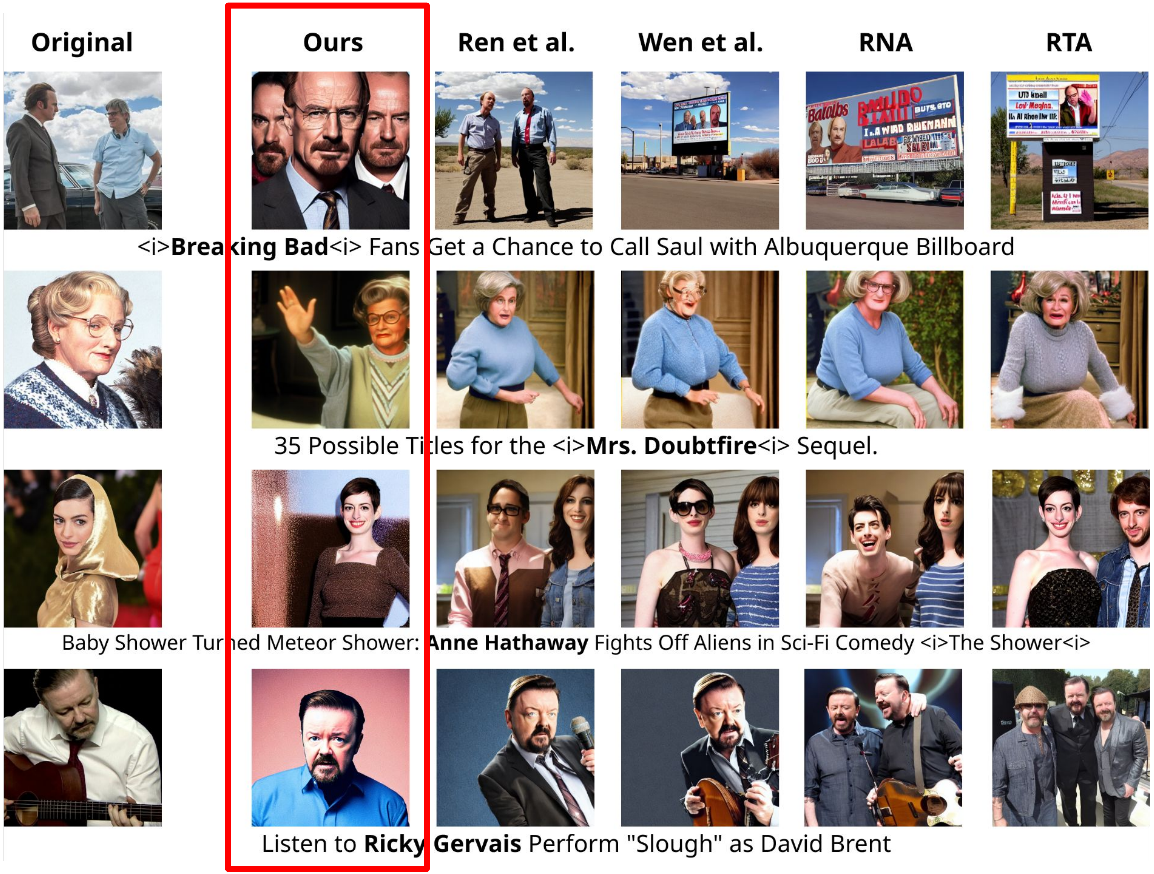
$$\left\| H_{\Delta\theta}(x_T)\, s_{\Delta\theta}(x_T) \right\|^2 - \alpha \log p_G(x_T)$$

Sharpness measure   Gaussian regularization

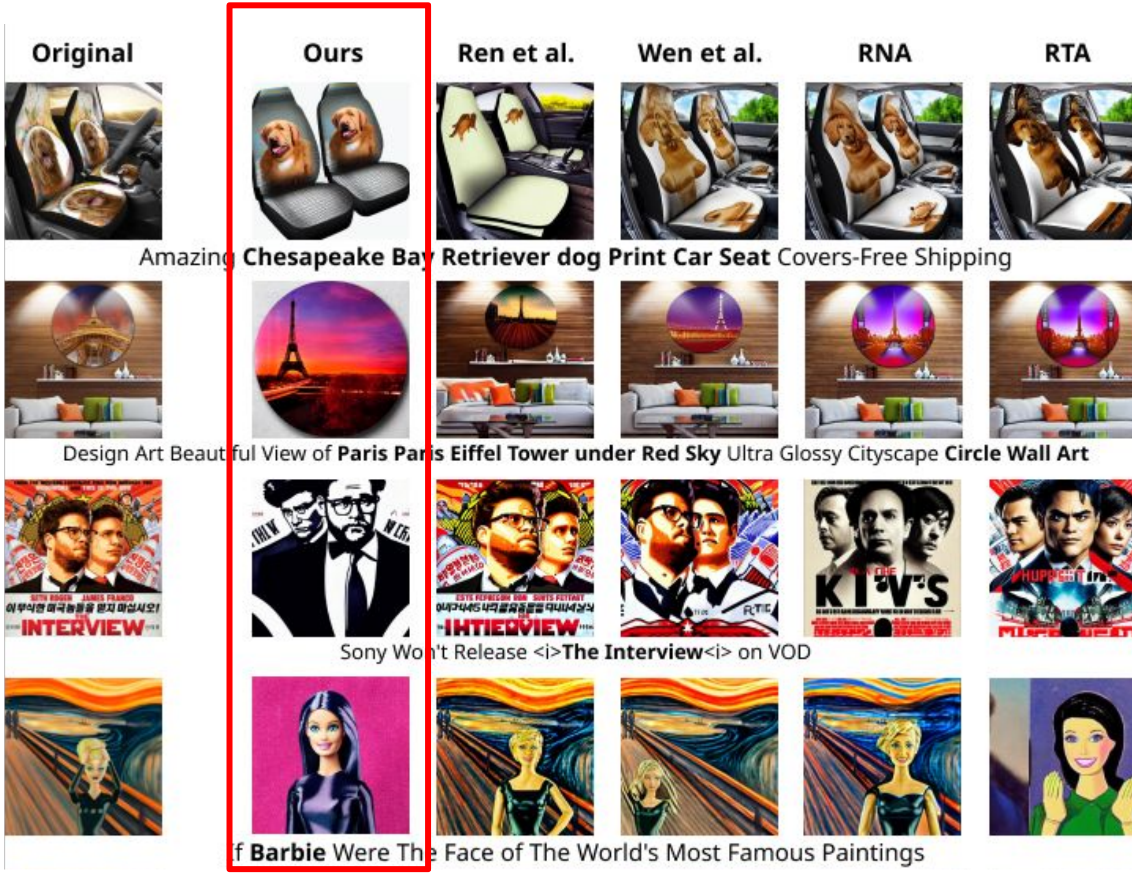
5. How can we mitigate it? (cont'd)



5. How can we mitigate it? (cont’d)



5. How can we mitigate it? (cont'd)



6. Advertisement

Visit:
https://github.com/Dongjae0324/sharpness_memorization_diffusion
and push “STAR”!