THE SUPERPOSITION OF DIFFUSION MODELS

using the Itô Density Estimator

Marta Skreta* ◦ Lazar Atanackovic* ◦ Joey Bose

Alexander Tong ◦ Kirill Neklyudov

We are in the “Cambrian Explosion” of diffusion models

IMAGES

Stable Diffusion

Midjourney

Imagen

DALL-E

AUDIO

VIDEOS

Stable Video

Luma AI

Sora

Pika Labs

ROBOTICS

WEATHER

PROTEINS & MOLECULES

AlphaFold 3

Chai-1

GenMol

RF*diffusion*

EDM

EvoDiff

DiffDock

MatterGen

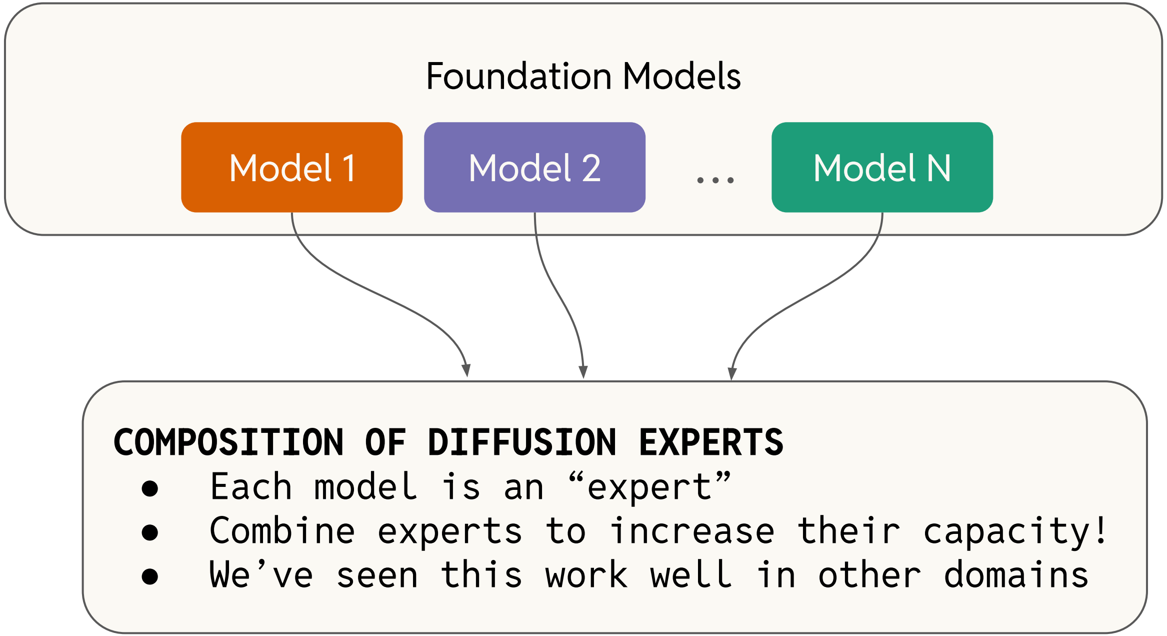
HEALTHCARE

How can we get the benefits of all these models?

Should we just keep training larger and larger models?

Not always possible!

What about combining multiple models together?

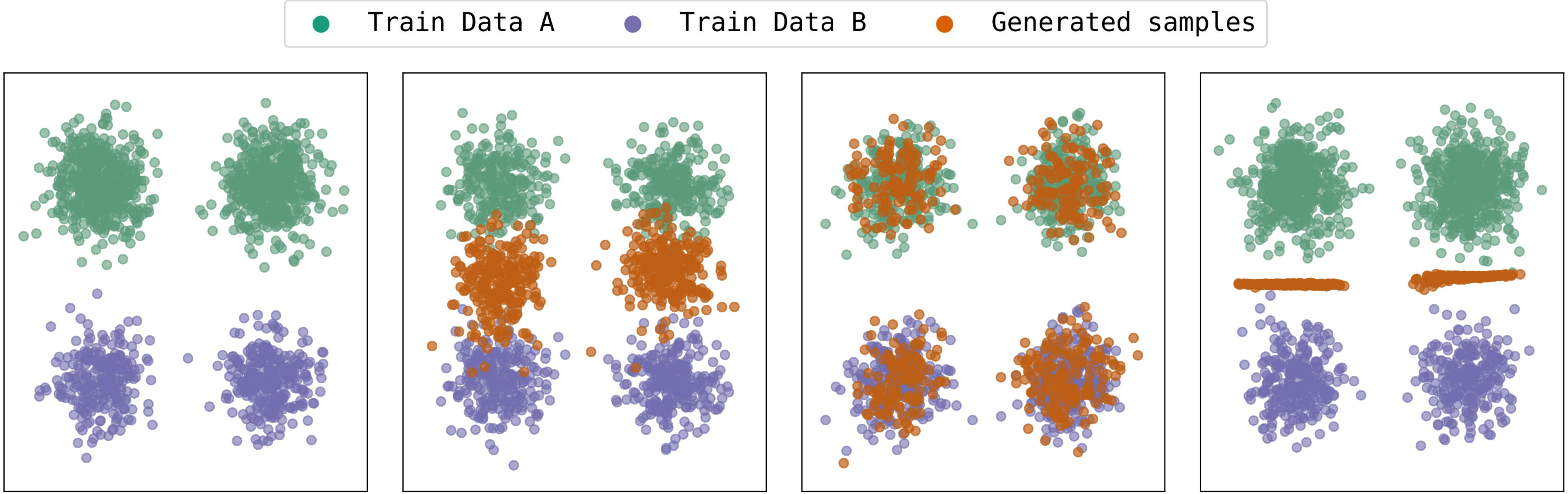


COMPOSITION OF DIFFUSION EXPERTS

- Each model is an “expert”
- Combine experts to increase their capacity!
- We’ve seen this work well in other domains

RESEARCH QUESTION:

Can we combine pre-trained diffusion models **solely at inference** in a theoretically sound and efficient manner?



Training Data

Averaging of model outputs[1]

[1] Liu et al. 2022

Sampling from a **mixture of densities** [ours]

logical **OR**

Sampling from **equal densities** [ours]

logical **AND**

THE SUPERPOSITION OF DIFFUSION MODELS (*SuperDiff*)

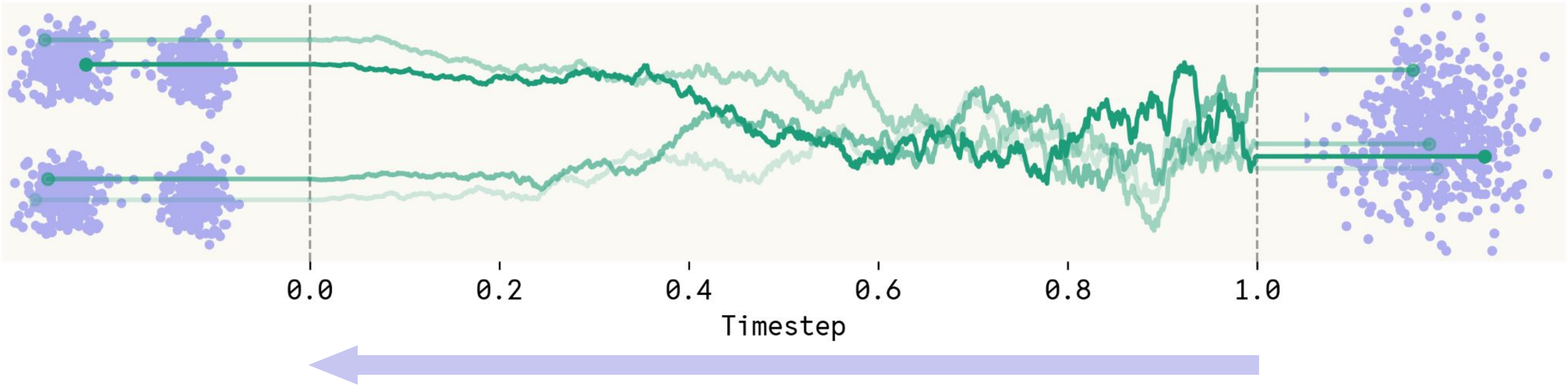
What are Diffusion Models?

Goal: generate samples from some data distribution $p_{\text{data}} \in \mathbb{P}(\mathbb{R}^d)$

Forward Process: $dx_t = f_t(x_t)dt + g_t dW_t, \quad x_0 \sim q_0(x_0)$
(data -> noise)

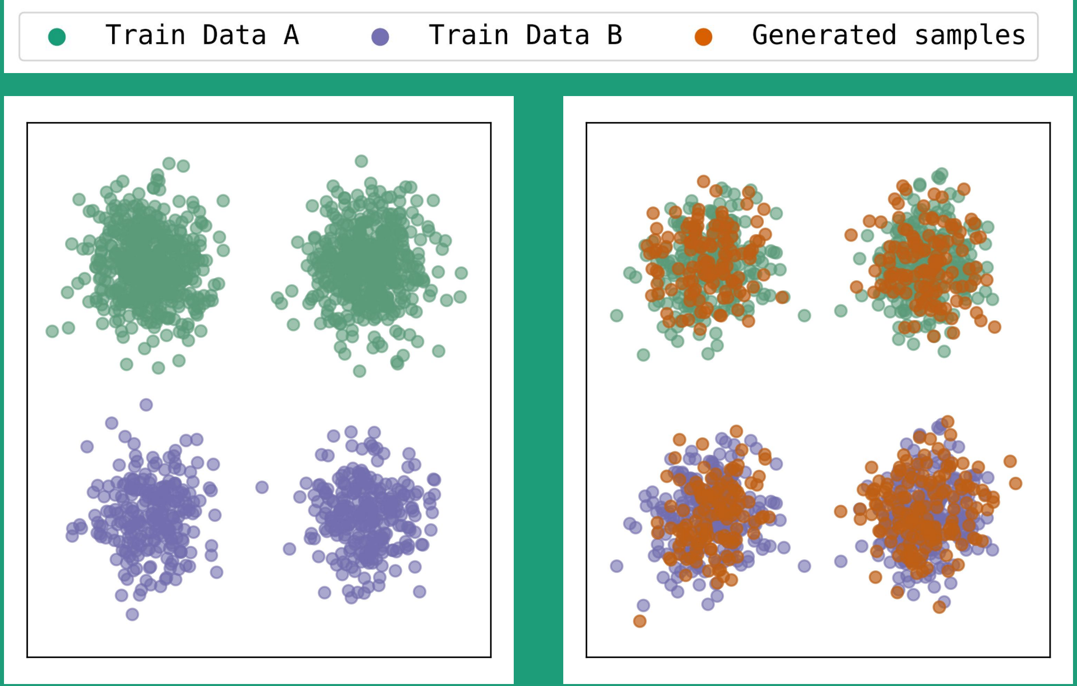
$p_{\text{data}} := q_0(x_0)$

$p_{\text{noise}} := q_1(x_1) = \mathcal{N}(x_1|0, I)$



Reverse Process: $dx_\tau = \underbrace{u_\tau(x_\tau)}_{\text{Vector field: } -f_t(x_\tau) + g_t^2 \boxed{\nabla \log q_t(x_\tau)}\ \text{Score (learned function)}} d\tau + g_t d\overline{W}_\tau \qquad x_{\tau=0} \sim q_1(x_0) \qquad \tau = 1 - t$
(noise -> data)

SuperDiff (OR)

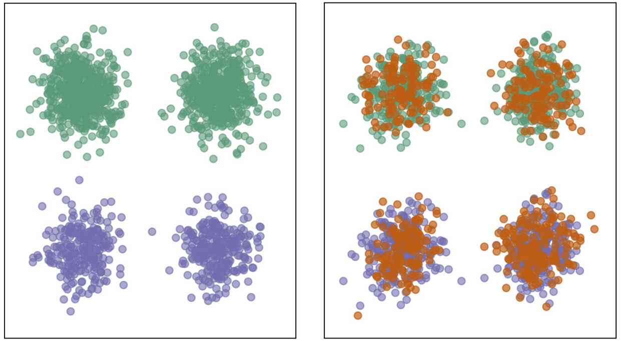


Training Data

Sampling from a **mixture of densities** [ours]

Sampling proportional to densities

`logical OR`



Given different processes and the corresponding vector fields:

$$\frac{\partial q_t^i(x)}{\partial t} = -\langle \nabla_x, q_t^i(x) u_t^i(x) \rangle + \frac{g_t^2}{2} \Delta q_t^i(x)\,, \quad i = 1, \dots, M$$

We want the model that samples from the mixture:

$$\frac{\partial q_t(x)}{\partial t} = -\langle \nabla_x, q_t^i(x) u_t^i(x) \rangle + \frac{g_t^2}{2} \Delta q_t^i(x), \quad q_t(x) = \frac{1}{M} \sum_{i=1}^{M} q_t^i(x)\,, \quad u_t(x) - ?$$

problem

Reverse SDE:

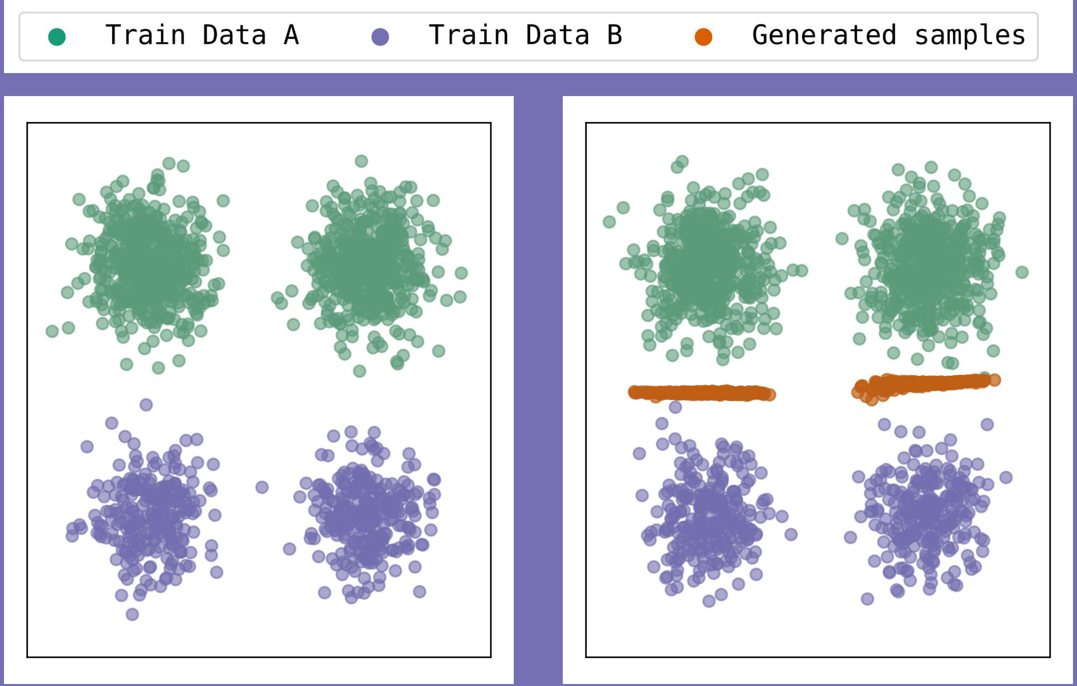
$$dx_\tau = u_\tau(x_\tau) d\tau + g_t d\overline{W}_\tau$$

where vector fields are weighted proportionally to densities:

$$u_\tau(x_\tau) = \sum_{i=1}^M \frac{q_t^i(x)}{\sum_j q_t^j(x)} u_\tau^i(x_\tau) \quad \texttt{for } M \texttt{ models}$$

solution

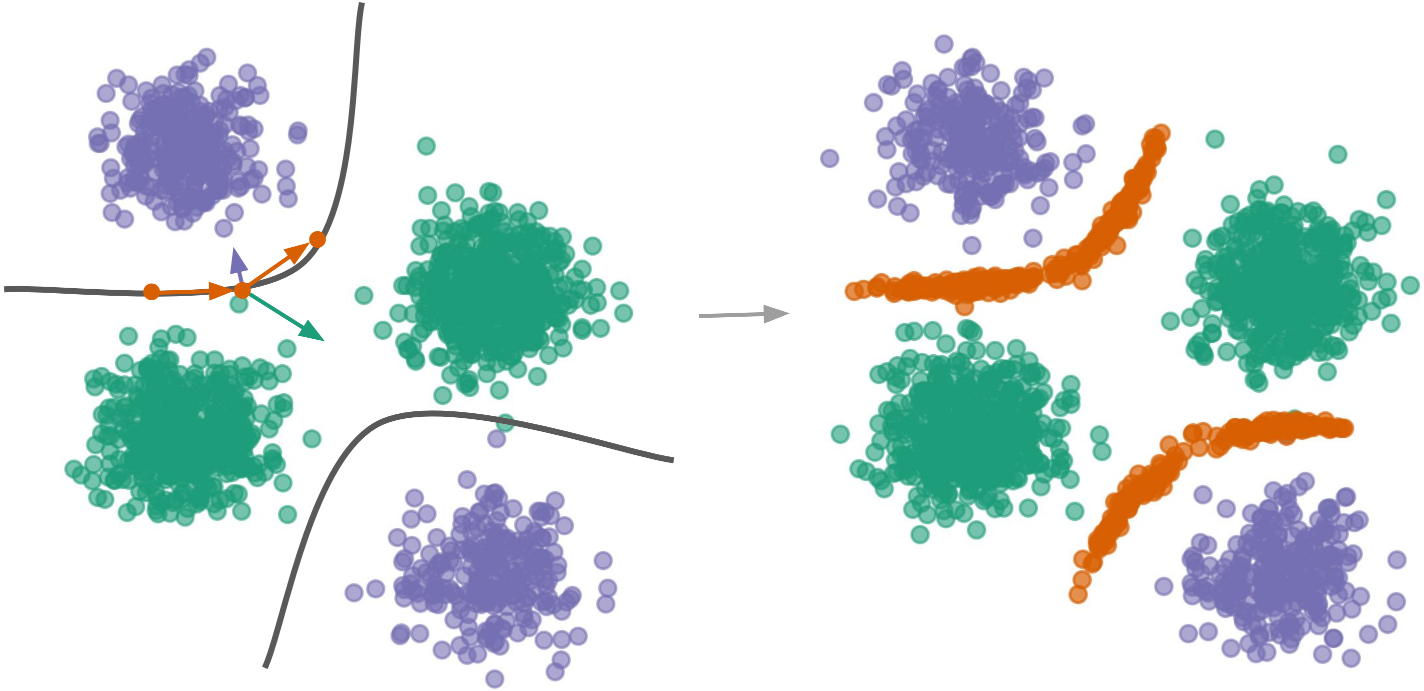
SuperDiff (AND)



Training Data

Sampling from **equal densities** [ours]

How do we sample from equal densities?

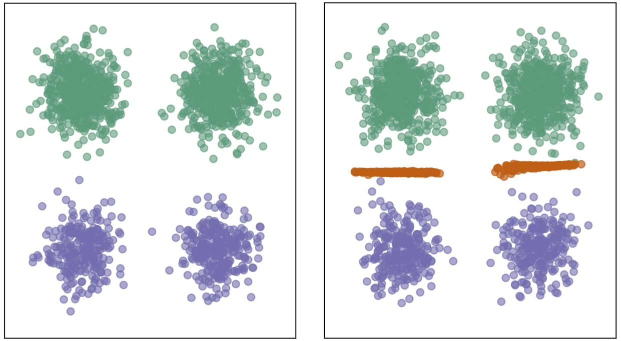


Reverse SDE:

$$dx_\tau = u_\tau(x_\tau)d\tau + g_t d\overline{W}_\tau$$

Sampling from equal densities

logical AND



Reverse SDE:

$$dx_\tau = u_\tau(x_\tau)d\tau + g_t d\overline{W}_\tau$$

Weight vector fields such that densities are equal:

$$q^1_{t+dt}(x_{t+dt}) = q^2_{t+dt}(x_{t+dt})$$

$$q^1_{1-\tau-dt}(x_\tau + (\textstyle\sum_j \kappa_j u_j(x_\tau))dt) = q^2_{1-\tau-dt}(x_\tau + (\textstyle\sum_j \kappa_j u_j(x_\tau))dt) \quad \text{such that} \quad \sum_j \kappa_j = 1$$

For two models:

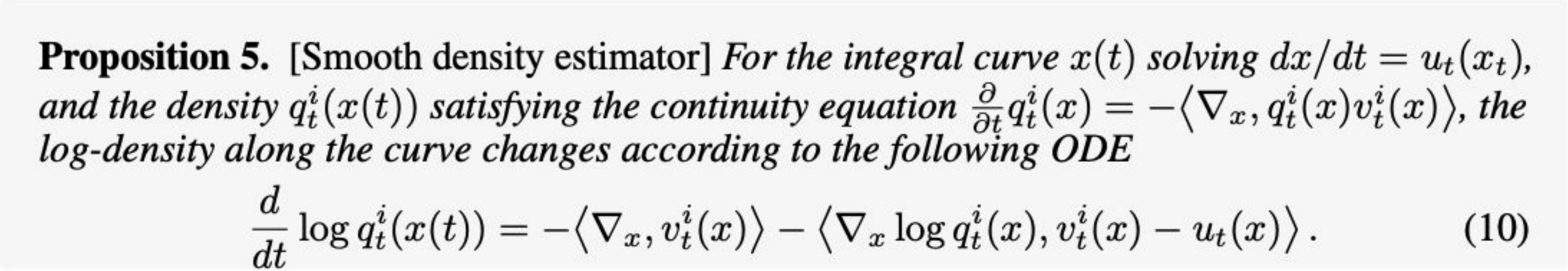
$$q^1_{1-\tau-dt}(x_\tau + (\kappa_1 u_1(x_\tau) + (1-\kappa_1)u_2(x_\tau))dt) = q^2_{1-\tau-dt}(x_\tau + (\kappa_1 u_1(x_\tau) + (1-\kappa_1)u_2(x_\tau))dt)$$

… or system of linear equations for j > 2 !

How can we get the densities???

ODE Case

Need: $$\log q_{1-\tau-dt}(x_\tau) = \log q_{1-\tau}(x_\tau) + \underset{\text{???}}{\boxed{d \log q_{1-\tau}(x_\tau)}}$$



Proposition 5. [Smooth density estimator] *For the integral curve $x(t)$ solving $dx/dt = u_t(x_t)$, and the density $q^i_t(x(t))$ satisfying the continuity equation $\frac{\partial}{\partial t} q^i_t(x) = -\langle \nabla_x, q^i_t(x) v^i_t(x) \rangle$, the log-density along the curve changes according to the following ODE*

$$\frac{d}{dt} \log q^i_t(x(t)) = -\langle \nabla_x, v^i_t(x) \rangle - \langle \nabla_x \log q^i_t(x), v^i_t(x) - u_t(x) \rangle \,. \tag{10}$$

$$dz(t, x(t)) = \underbrace{\frac{\partial}{\partial t} z(t, x(t)) dt}_{\text{change in time}} + \underbrace{\frac{\partial}{\partial x} z(t, x(t)) dx}_{\text{change in position}}$$

SDE Case

Need: $\log q_{1-\tau-dt}(x_\tau) = \log q_{1-\tau}(x_\tau) + d\log q_{1-\tau}(x_\tau)$

SDE gives us an extra noise term – can't ignore!

$$dz(t,x(t)) = \frac{\partial}{\partial t}z(t,x(t))dt + \frac{\partial}{\partial x}z(t,x(t))dx + \underbrace{\frac{1}{2}\frac{\partial^2}{\partial x^2}z(t,x(t))(dx)^2}_{\text{Due to noise! } (d\overline{W})^2 = O(dt)}$$

(Itô's Lemma!)

$$d\log q_{1-\tau}(x_\tau) = \underbrace{\frac{\partial}{\partial \tau}\log q_{1-\tau}(x_\tau)d\tau}_{\text{Fokker-Planck equation}} + \underbrace{\frac{\partial}{\partial x}\log q_{1-\tau}(x_\tau)}_{\text{Score: } \nabla \log q_{1-\tau}(x_\tau)}dx_\tau + \frac{1}{2}\underbrace{\frac{\partial^2}{\partial x^2}\log q_{1-\tau}(x_\tau)}_{\text{Laplacian: } \Delta \log q_{1-\tau}(x_\tau)}(dx_\tau)^2$$

Reverse step: $u_t(x_\tau)d\tau + g_t d\overline{W}_\tau$

✨ Itô Density Estimator ✨

Need: $\log q_{1-\tau-dt}(x_\tau) \ = \log q_{1-\tau}(x_\tau) + d\log q_{1-\tau}(x_\tau)$

$$d\log q_{1-\tau}(x_\tau) = \underbrace{\frac{\partial}{\partial\tau}\log q_{1-\tau}(x_\tau)d\tau}_{\text{Fokker-Planck equation}} + \nabla\log q_{1-\tau}(x_\tau)u_t(x_\tau)d\tau \boxed{\cancel{+\frac{1}{2}\Delta\log q_{1-\tau}(x_\tau)(g_t^2)\underbrace{(d\overline{W})^2}_{(d\overline{W})^2 = d\tau}}}$$

$$\frac{\partial}{\partial\tau}\log q_{1-\tau}(x_\tau)d\tau = (\langle\nabla, f_{1-\tau}(x_\tau)\rangle + \langle\nabla\log q_{1-\tau}(x_\tau), f_{1-\tau}(x_\tau)\rangle)d\tau - \boxed{\cancel{-\frac{g_{1-\tau}^2}{2}\Delta\log q_{1-\tau}(x_\tau)d\tau}} - \frac{g_{1-\tau}^2}{2}||\nabla\log q_{1-\tau}(x_\tau)||^2 d\tau$$

No more Laplacian! Density estimation with no extra cost!

✨ Itô Density Estimator ✨

Need: $\log q_{1-\tau-dt}(x_\tau) \;=\; \log q_{1-\tau}(x_\tau) + d\log q_{1-\tau}(x_\tau)$

$$d\log q_{1-\tau}(x_\tau) = \underbrace{\frac{\partial}{\partial\tau}\log q_{1-\tau}(x_\tau)d\tau}_{\text{Fokker-Planck equation}} + \nabla\log q_{1-\tau}(x_\tau)u_t(x_\tau)d\tau \cancel{\boxed{+\frac{1}{2}\Delta\log q_{1-\tau}(x_\tau)(g_t^2)\underbrace{(d\overline{W})^2}_{(d\overline{W})^2 = d\tau}}}$$

$$\frac{\partial}{\partial\tau}\log q_{1-\tau}(x_\tau)d\tau = (\langle\nabla, f_{1-\tau}(x_\tau)\rangle + \langle\nabla\log q_{1-\tau}(x_\tau), f_{1-\tau}(x_\tau)\rangle)d\tau -$$

$$\cancel{\boxed{-\frac{g_{1-\tau}^2}{2}\Delta\log q_{1-\tau}(x_\tau)d\tau}} - \frac{g_{1-\tau}^2}{2}||\nabla\log q_{1-\tau}(x_\tau)||^2 d\tau$$

$$d\log q_{1-\tau}(x_\tau) = \langle dx_\tau, \nabla\log q_{1-\tau}(x_\tau)\rangle + \left(\langle\nabla, f_{1-\tau}(x_\tau)\rangle + \left\langle u_{1-\tau}(x_\tau), \nabla\log q_{1-\tau}(x_\tau)\right\rangle\right)d\tau$$

✨ Itô Density Estimator ✨

Need: $\log q_{1-\tau-dt}(x_\tau) = \log q_{1-\tau}(x_\tau) + d\log q_{1-\tau}(x_\tau)$

$$d\log q_{1-\tau}(x_\tau) = \frac{\partial}{\partial\tau}\log q_{1-\tau}(x_\tau)d\tau + \nabla \log q_{1-\tau}(x_\tau)u_t(x_\tau)d\tau + \frac{1}{2}\Delta \log q_{1-\tau}(x_\tau)(g_t^2)(d\overline{W})^2$$

This can be anything!!! Not just vector field of reverse dynamics!!!

Expand & collect terms:

$$d\log q_{1-\tau}(x_\tau) = \langle \boxed{dx_\tau}, \nabla \log q_{1-\tau}(x_\tau)\rangle + \left(\langle \nabla, f_{1-\tau}(x_\tau)\rangle + \left\langle \boxed{u_{1-\tau}(x_\tau),} \nabla \log q_{1-\tau}(x_\tau)\right\rangle\right)d\tau$$

Using this density estimator is ~5x faster than computing div. & 30% less memory on image experiments

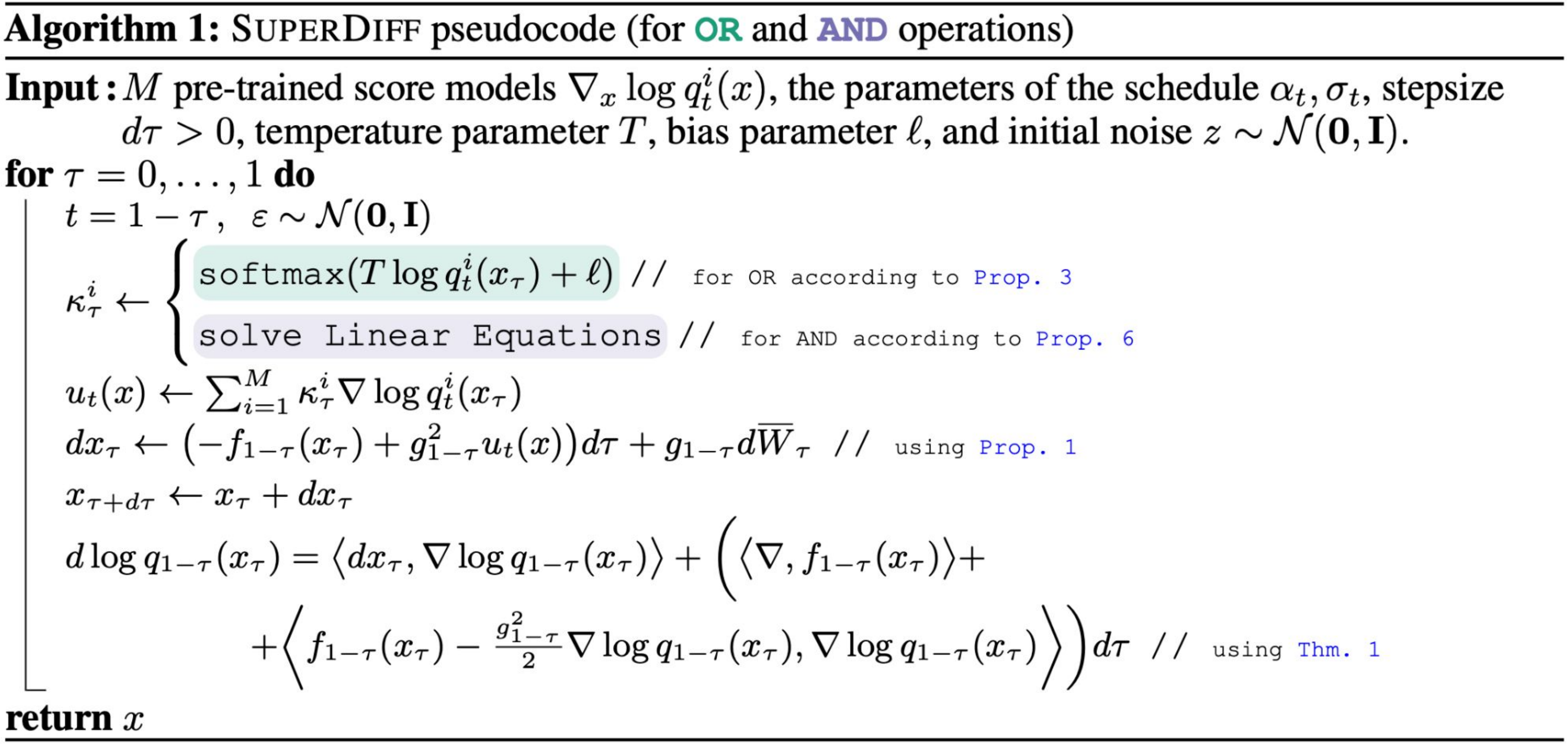
Some notes on SuperDiff

BENEFITS ✅

- Principled approach (continuity equation is satisfied)
- Architecture-agnostic
- Can work for any number of models

CAUTION ⚠️

- For exact density computation, assumption is that learned score is true score



Algorithm 1: SuperDiff pseudocode (for OR and AND operations)

```
Input : $M$ pre-trained score models $\nabla_x \log q_t^i(x)$, the parameters of the schedule $\alpha_t, \sigma_t$, stepsize
        $d\tau > 0$, temperature parameter $T$, bias parameter $\ell$, and initial noise $z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
for $\tau = 0, \ldots, 1$ do
    $t = 1 - \tau\,,\ \ \varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    $\kappa_\tau^i \leftarrow \begin{cases} \texttt{softmax}(T \log q_t^i(x_\tau) + \ell) & \text{// for OR according to Prop. 3} \\ \texttt{solve Linear Equations} & \text{// for AND according to Prop. 6} \end{cases}$
    $u_t(x) \leftarrow \sum_{i=1}^M \kappa_\tau^i \nabla \log q_t^i(x_\tau)$
    $dx_\tau \leftarrow \big(-f_{1-\tau}(x_\tau) + g_{1-\tau}^2 u_t(x)\big)d\tau + g_{1-\tau} d\overline{W}_\tau$   // using Prop. 1
    $x_{\tau+d\tau} \leftarrow x_\tau + dx_\tau$
    $d \log q_{1-\tau}(x_\tau) = \langle dx_\tau, \nabla \log q_{1-\tau}(x_\tau)\rangle + \bigg(\langle \nabla, f_{1-\tau}(x_\tau)\rangle +$
        $+ \Big\langle f_{1-\tau}(x_\tau) - \frac{g_{1-\tau}^2}{2} \nabla \log q_{1-\tau}(x_\tau), \nabla \log q_{1-\tau}(x_\tau) \Big\rangle \bigg) d\tau$   // using Thm. 1
return $x$
```

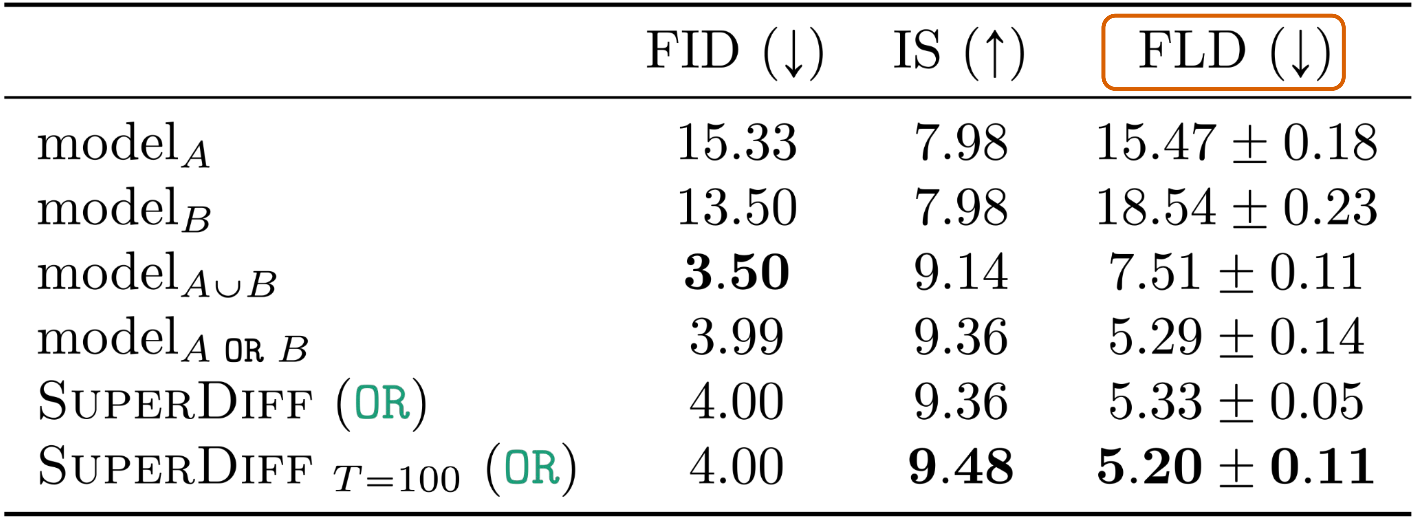
EXPERIMENTS

[1] Unconditional Image Generation: *Validating the Method*

CIFAR-10

- Divide CIFAR-10 training set into two halves:
  - Part A: First 5 labels
  - Part B: Last 5 labels
- Train separate diffusion model on each half

Novelty, fidelity, & diversity



	FID (↓)	IS (↑)	FLD (↓)
model$_A$	15.33	7.98	15.47 ± 0.18
model$_B$	13.50	7.98	18.54 ± 0.23
model$_{A \cup B}$	**3.50**	9.14	7.51 ± 0.11
model$_{A \texttt{ OR } B}$	3.99	9.36	5.29 ± 0.14
SuperDiff (OR)	4.00	9.36	5.33 ± 0.05
SuperDiff $_{T=100}$ (OR)	4.00	**9.48**	$\mathbf{5.20 \pm 0.11}$

[2] Unconditional Protein Generation

Q: I have two models that generate proteins using different datasets and/or architectures – how do I combine them?

Goal: Generate 3D coordinates of protein backbones

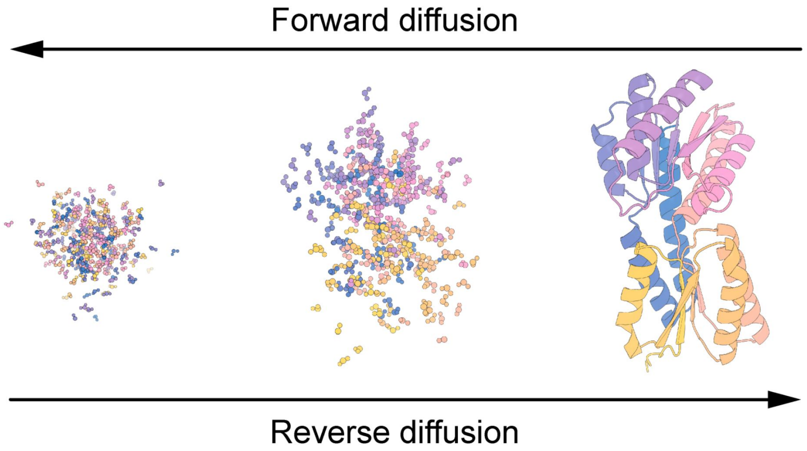


Image from Proteus

Two protein models:

- **Proteus**[1] (more designable)
- **FrameDiff**[2] (more diverse)

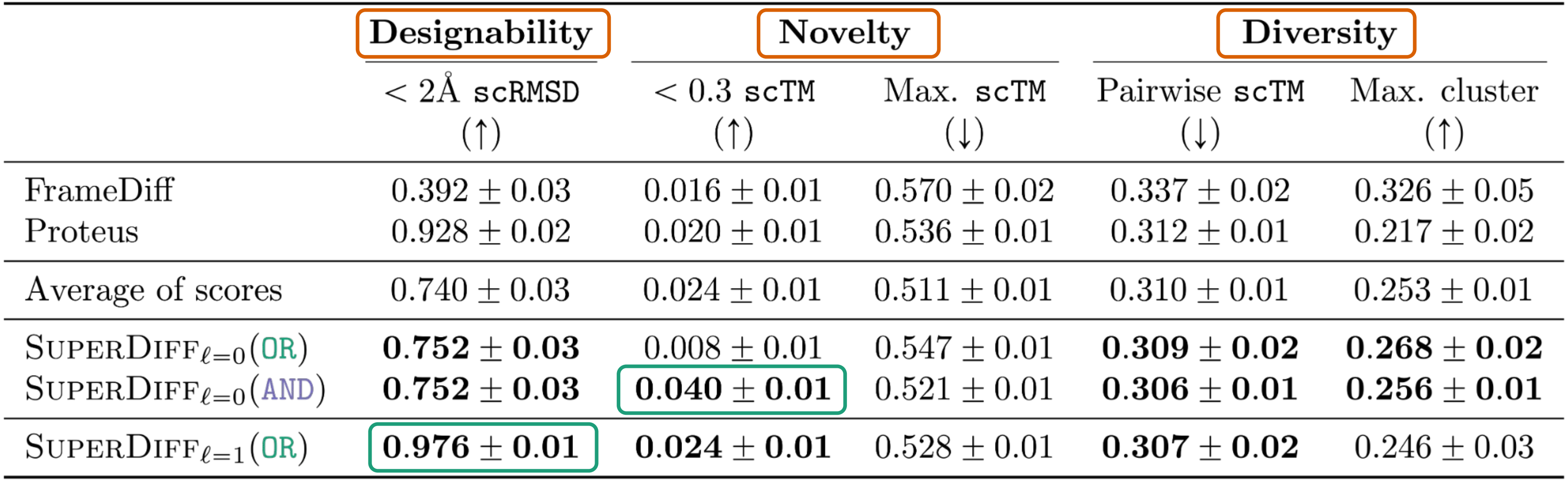
[1] Wang et al. ICML, 2024.
[2] Yim et al. ICML, 2023.

Goal: Generate 3D coordinates of protein backbones

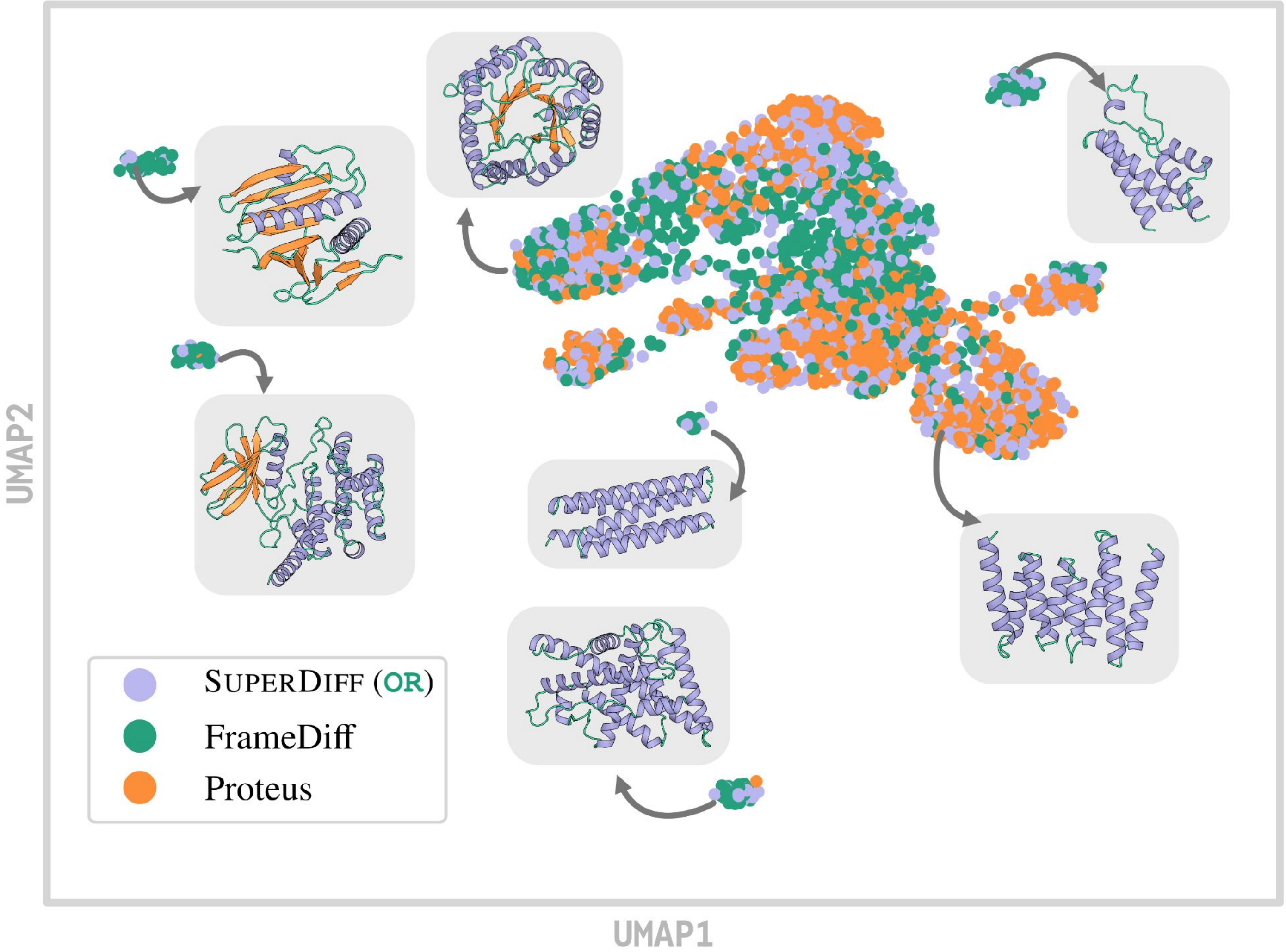
Is there a sequence that folds into the structure?

How similar is the protein to other sequences in the training data?

How similar are the proteins to each other?



	Designability	Novelty		Diversity	
	< 2Å scRMSD (↑)	< 0.3 scTM (↑)	Max. scTM (↓)	Pairwise scTM (↓)	Max. cluster (↑)
FrameDiff	0.392 ± 0.03	0.016 ± 0.01	0.570 ± 0.02	0.337 ± 0.02	0.326 ± 0.05
Proteus	0.928 ± 0.02	0.020 ± 0.01	0.536 ± 0.01	0.312 ± 0.01	0.217 ± 0.02
Average of scores	0.740 ± 0.03	0.024 ± 0.01	0.511 ± 0.01	0.310 ± 0.01	0.253 ± 0.01
SuperDiff$_{\ell=0}$(OR)	$\mathbf{0.752 \pm 0.03}$	0.008 ± 0.01	0.547 ± 0.01	$\mathbf{0.309 \pm 0.02}$	$\mathbf{0.268 \pm 0.02}$
SuperDiff$_{\ell=0}$(AND)	$\mathbf{0.752 \pm 0.03}$	$\mathbf{0.040 \pm 0.01}$	0.521 ± 0.01	$\mathbf{0.306 \pm 0.01}$	$\mathbf{0.256 \pm 0.01}$
SuperDiff$_{\ell=1}$(OR)	$\mathbf{0.976 \pm 0.01}$	$\mathbf{0.024 \pm 0.01}$	0.528 ± 0.01	$\mathbf{0.307 \pm 0.02}$	0.246 ± 0.03


SuperDiff (OR)
FrameDiff
Proteus
UMAP1
UMAP2

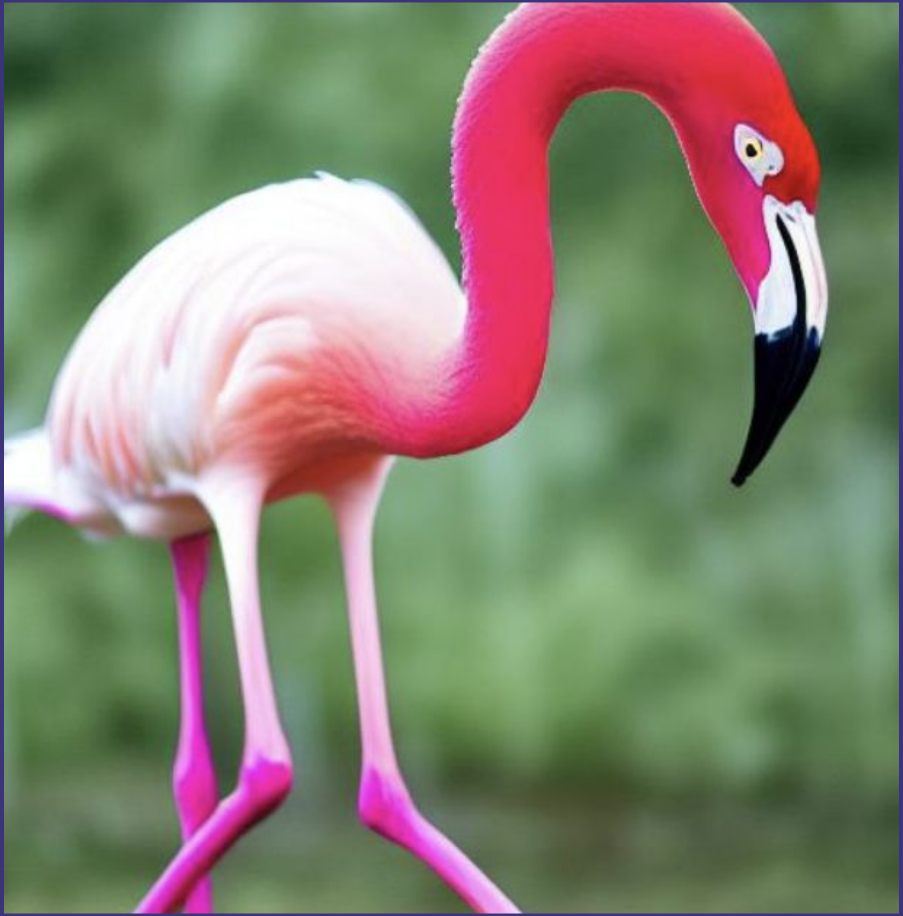
[3] Conditional Image Generation

Q: How would you generate an image of a flamingo that looks like a candy cane?

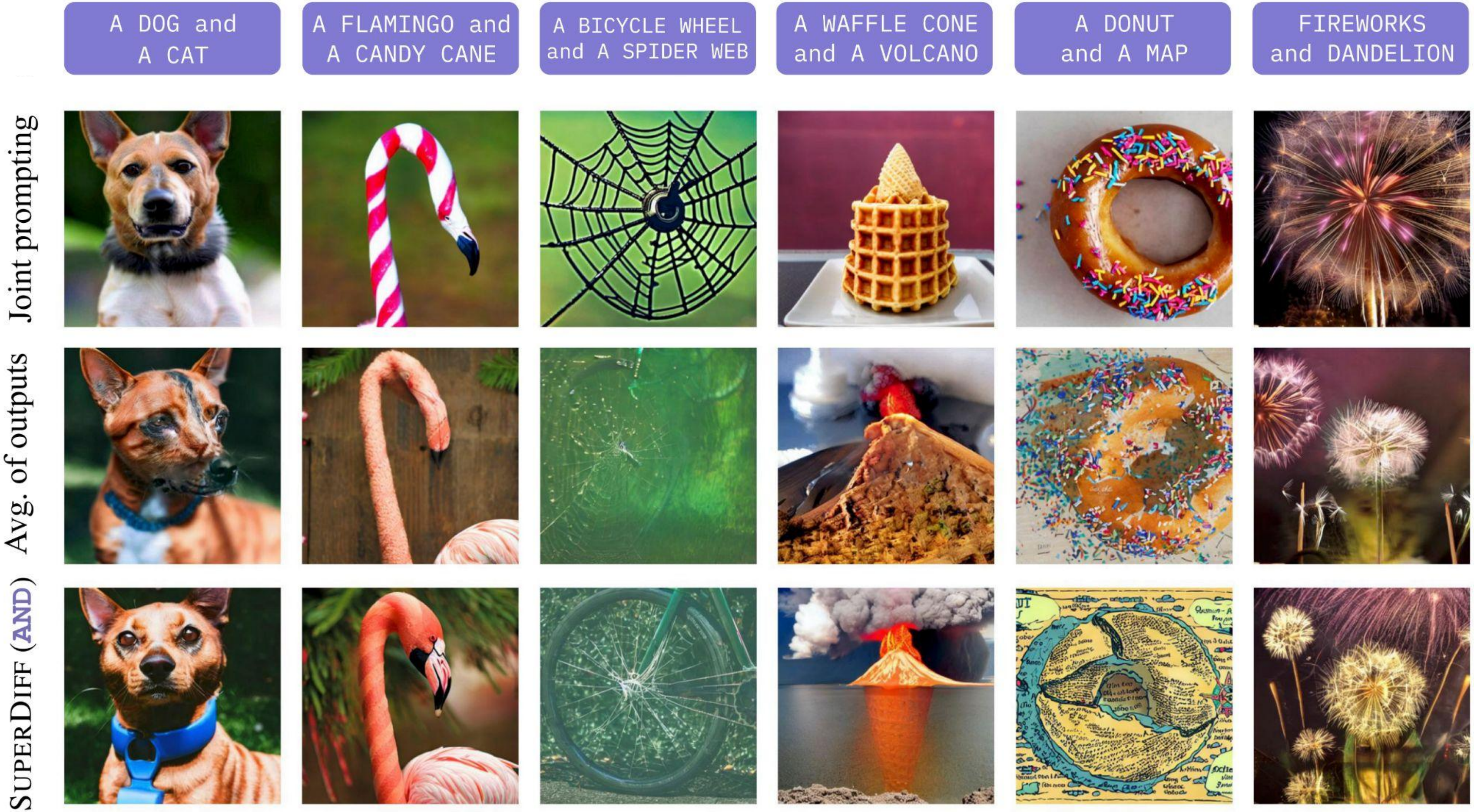
🤗 Hugging Face

Stable Diffusion V1.4

Prompt: *“A flamingo that looks like a candy cane.”*



Stable Diffusion v1.4



A DOG and A CAT

A FLAMINGO and A CANDY CANE

A BICYCLE WHEEL and A SPIDER WEB

A WAFFLE CONE and A VOLCANO

A DONUT and A MAP

FIREWORKS and DANDELION

Joint prompting

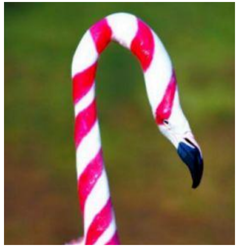
Avg. of outputs

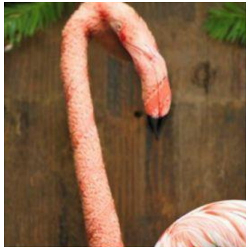
SUPERDIFF (AND)

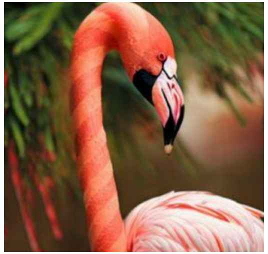
*Joint prompting: “A OBJECT_1 that looks like a OBJECT_2.”

Stable Diffusion v1.4

	Cosine sim. of text & img embeds	Human preference alignment	LLM-based QA
	Min. CLIP(↑)	Min. ImageReward (↑)	Min. TIFA (↑)
Joint prompting	23.87	−1.62	27.58
Average of scores	24.23	−1.57	32.48
SuperDiff (AND)	**24.79**	**−1.39**	**39.92**


Joint prompting


Avg. of outputs


SuperDiff(AND)

Prompt 1: “This image looks like a flamingo.”
Prompt 2: “This image looks like a candy cane.”

[4] Conditional Molecule Generation

Q: How would you generate a molecule that:

- inhibits the enzyme GSK3β and
- has drug-likeness?

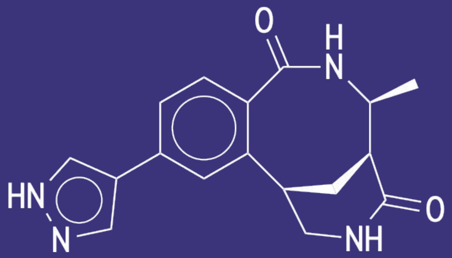
Model: LDMol

Prompt	GSK3β inhibition % (↑)	Drug-likeness % (↑)	Product of Properties (↑)	Val & Uniq %
“This molecule inhibits GSK3β.”	**0.411 ± 0.034**	0.266 ± 0.058	0.107 ± 0.024	0.74
“This molecule looks like a drug.”	0.033 ± 0.011	**0.884 ± 0.008**	0.030 ± 0.010	0.83
“This molecule inhibits GSK3β and looks like a drug.”	0.287 ± 0.029	0.631 ± 0.050	0.171 ± 0.029	0.74
AVERAGE OF SCORES	0.287 ± 0.014	0.580 ± 0.020	0.154 ± 0.012	0.77
SuperDiff (AND)	0.277 ± 0.024	0.668 ± 0.027	**0.177 ± 0.024**	0.78

[1] Chang and Ye, 2024.

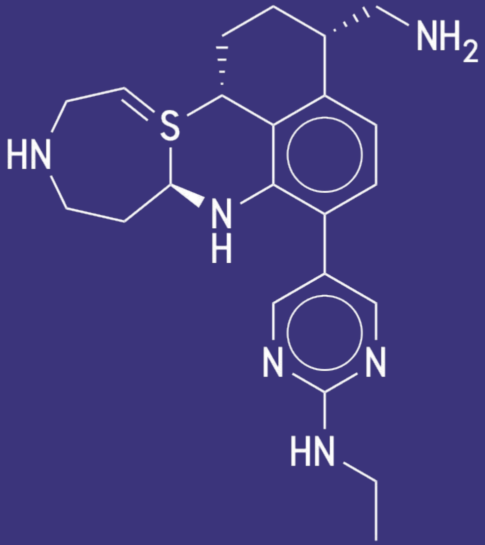
Model: LDMol

Joint prompting



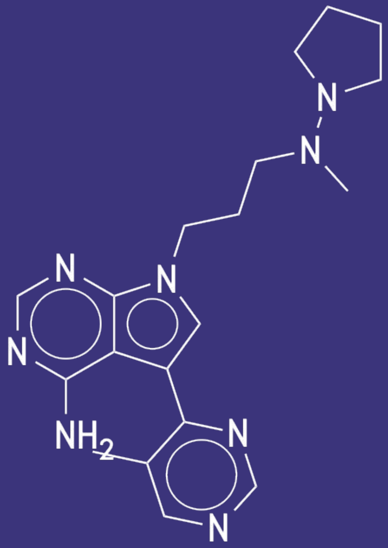
GSK3β: 0.39
QED: 0.74

Average of Scores



GSK3β: 0.40
QED: 0.58

SuperDiff(AND)



GSK3β: 0.53
QED: 0.71

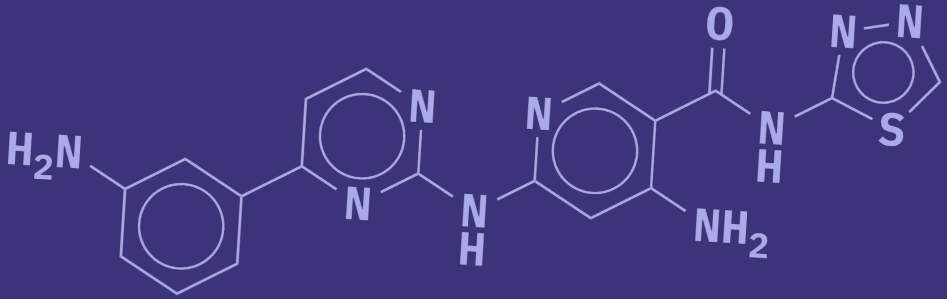
[1] Chang and Ye, 2024.

Q: How would you generate a molecule that:

- inhibits the enzyme JNK3 **AND**
- inhibits the enzyme GSK3β?

Model: LDMol

Prompt	Min(JNK3, GSK3β) (↑)	Product of Properties (↑)
“This molecule inhibits JNK3.”	0.135 ± 0.011	0.057 ± 0.007
“This molecule inhibits GSK3β.”	0.183 ± 0.014	0.056 ± 0.011
“This molecule inhibits JNK3 and inhibits GSK3β.”	0.186 ± 0.045	0.071 ± 0.022
AVERAGE OF SCORES	0.199 ± 0.012	0.073 ± 0.013
SuperDiff(AND)	**0.209 ± 0.035**	**0.080 ± 0.025**



Top-1 SuperDiff (AND)

JNK3: 0.52

GSK3β: 0.69

[1] Chang and Ye, 2024.

arXiv

SuperDiff!

GitHub

Marta
Skreta*

Lazar
Atanackovic*

Joey
Bose

Alexander
Tong

Kirill
Neklyudov

University of Toronto ◦ Vector Institute ◦ University of Oxford ◦ Mila – Québec AI Institute ◦ Université de Montréal