Continuous Generative Models

Andrew Valentini

University of Florida

July 2026

Materials Generation

General Goal:

- Learn $\mathcal{M}(\boldsymbol{A}, \boldsymbol{F}, \boldsymbol{L})$ and draw new samples $\boldsymbol{x} \sim \mathcal{M}$ (de novo generation)

Specific Goal:

- De novo generation for superconductors
  - Hopefully ordered and high T_c

... do this with **continuous generative models**

Representation of Materials

We express a unit cell of N atoms as $\mathcal{M}(\boldsymbol{A}, \boldsymbol{F}, \boldsymbol{L})$ where

- $\boldsymbol{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_N] \in \mathbb{R}^{118 \times N}$ are one-hot encodings of atomic type at each site
- $\boldsymbol{F} = [\boldsymbol{f}_1, \boldsymbol{f}_2, \ldots, \boldsymbol{f}_N] \in [0, 1)^{3 \times N}$ are fractional coordinates of the atoms
  - $p(\boldsymbol{F}_0)$ must satisfy translation symmetry invariance due to crystal periodicity
- $\boldsymbol{L} = [\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \boldsymbol{\ell}_3] \in \mathbb{R}^{3 \times 3}$ are the basis vectors of unit cell
  - $p(\boldsymbol{L}_0)$ must satisfy $E(3)$ equivariance; rotation, translation, and reflection symmetry
- These massive spaces make knowing $\mathcal{M}$ implicitly practically impossible
- Use continuous generative models to sample from a learned, approximate form of $\mathcal{M}$

Continuous Generative Models

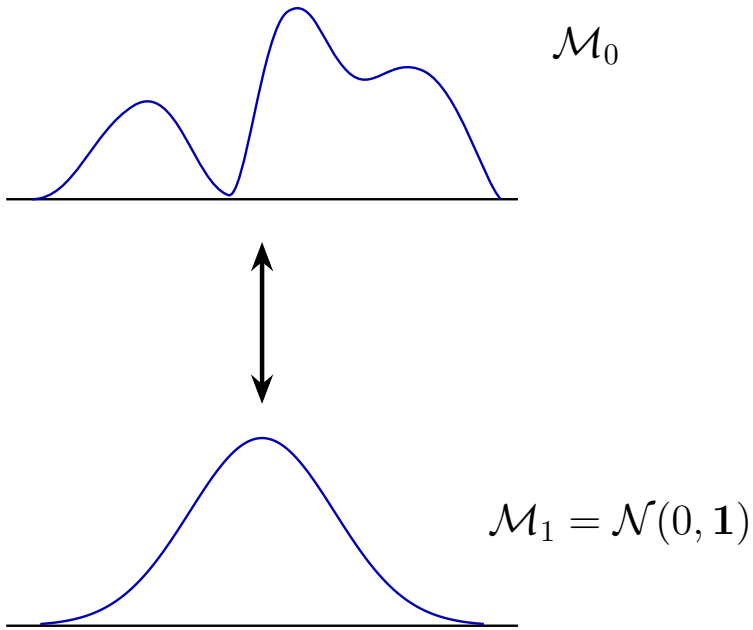
General framework for generating new samples from an unknown *full* distribution $\mathcal{M}$:

1. Let a neural net map $\mathcal{M}_0 \to \mathcal{N}(0, \mathbb{1})$ ($\mathcal{M}_0$ made up of known samples)
2. Have it learn the reverse process

Three such models:

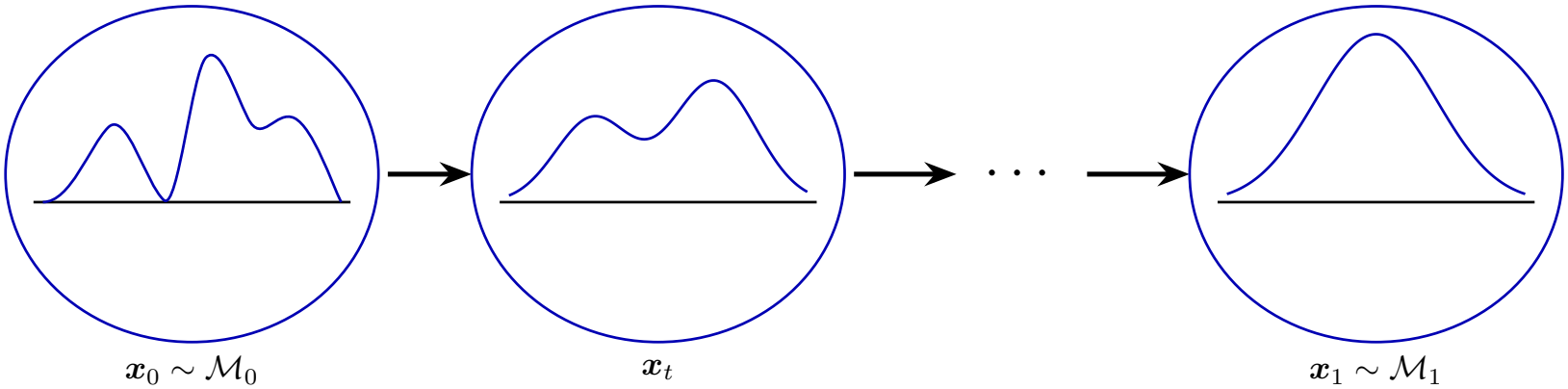
- Diffusion
- Flow Matching
- Stochastic Interpolants

(broadly used)



Diffusion Generally [1]

- Carry out **forward diffusion**, $\mathcal{M}_0 \to \mathcal{M}_1$, by gradually adding noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbb{1})$ to $\mathcal{M}_0$



- For a random step $t \sim U(0, 1)$, sample $\mathbf{x}_0 \sim \mathcal{M}_0$ by

$$\mathbf{x}_{t+1} = a_t \mathbf{x}_t + b_t \boldsymbol{\epsilon}_t$$

Determining Constants a_t and b_t

With $\boldsymbol{x}_{t+1} = a_t \boldsymbol{x}_t + b_t \boldsymbol{\epsilon}_t$, determine a_t & b_t by forcing $\text{Var}[\boldsymbol{x}_t] = 1$
...reminder that $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbb{1})$

$$\begin{aligned} \implies \text{Var}[\boldsymbol{x}_{t+1}] &= \text{Var}[a_t \boldsymbol{x}_t] + \text{Var}[b_t \boldsymbol{\epsilon}_t] \\ &= a_t^2 \text{Var}[\boldsymbol{x}_t] + b_t^2 \text{Var}[\boldsymbol{\epsilon}_t] \\ \implies 1 &= a_t^2 + b_t^2 \end{aligned}$$

Define $a_t = \sqrt{\alpha_t}$, then

$$\boldsymbol{x}_{t+1} = \sqrt{\alpha_t} \boldsymbol{x}_t + \sqrt{1 - \alpha_t^2} \boldsymbol{\epsilon}_t$$

$\alpha_t \equiv$ **noise scheduler**, defined such that $\alpha_1 = 0$ to ensure $\boldsymbol{x}_t \to \mathcal{N}(0, \mathbb{1})$ as $t \to 1$

Diffusion Learning Generally

With the forward diffusion process

$$\boldsymbol{x}_{t+1} = \sqrt{\alpha_t}\boldsymbol{x}_t + \sqrt{1-\alpha_t^2}\boldsymbol{\epsilon}_t,$$

let N.N. predict the cumulative noise $\boldsymbol{\epsilon}$ added to a sample $\boldsymbol{x}_0$ at a specific timestep t, $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$, and compare against known $\boldsymbol{\epsilon}$ (averaged over all $t, \boldsymbol{x}_0, \epsilon$):

$$\mathcal{L} = \mathbb{E}_{t,\boldsymbol{x}_0,\boldsymbol{\epsilon}}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\|^2\right]$$

Then invert the diffusion process ($\boldsymbol{x}_{t+1} \to \boldsymbol{x}_t$) to start from a known noise distribution $\mathcal{M}_1 \sim \mathcal{N}(0, \mathbb{1})$ and generate new samples in $\mathcal{M}_0$ (DNG)

Diffusion steps can also be written as $q(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t+1}; \mu, \sigma)$:

$$q(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t+1}; \sqrt{\alpha_t}\boldsymbol{x}_t, (1-\alpha_t^2)\mathbb{1})$$

To sample from $\mathcal{M}(\boldsymbol{A}, \boldsymbol{F}, \boldsymbol{L})$ for de novo generation, we diffuse the 3 variables separately and construct:

$$\mathcal{L}_{\mathcal{M}} = \lambda_{\mathbf{A}}\mathcal{L}_{\mathbf{A}} + \lambda_{\mathbf{F}}\mathcal{L}_{\mathbf{F}} + \lambda_{\mathbf{L}}\mathcal{L}_{\mathbf{L}}$$

- Diffusion for $\boldsymbol{X} = [\boldsymbol{A}, \boldsymbol{L}]$:

$$q(\boldsymbol{X}_t, \boldsymbol{X}_0) = \mathcal{N}(\boldsymbol{X}_t | \sqrt{\alpha_t}\boldsymbol{X}_0, (1 - \alpha_t^2)\mathbb{1})$$
$$\& \ \mathcal{L}_{\boldsymbol{X}} = \mathbb{E}_{\boldsymbol{\epsilon}_{\boldsymbol{X}}, t}\left[\left\|\boldsymbol{\epsilon}_{\boldsymbol{X}} - \hat{\boldsymbol{\epsilon}}_{\boldsymbol{X}}\right\|^2\right]$$

  where $E(3)$ symmetry for $\boldsymbol{L}$ is enforced in the transitions $\mathbf{x}_t \to \mathbf{x}_{t+1}$
- Diffusion for $\boldsymbol{F}$ is more complicated to enforce translational symmetry

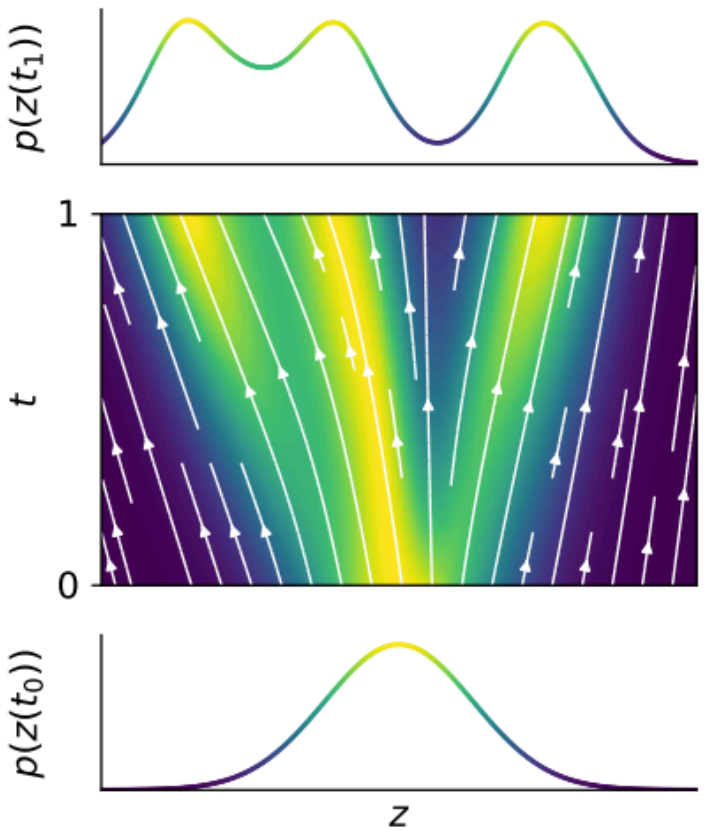
Flow Matching Generally [3]

Idea: Replace diffusion's stochastic process by deterministic mapping $\boldsymbol{v_\theta}$ (much fewer steps required)

- Learned $\boldsymbol{v_\theta}$ then generates new samples by

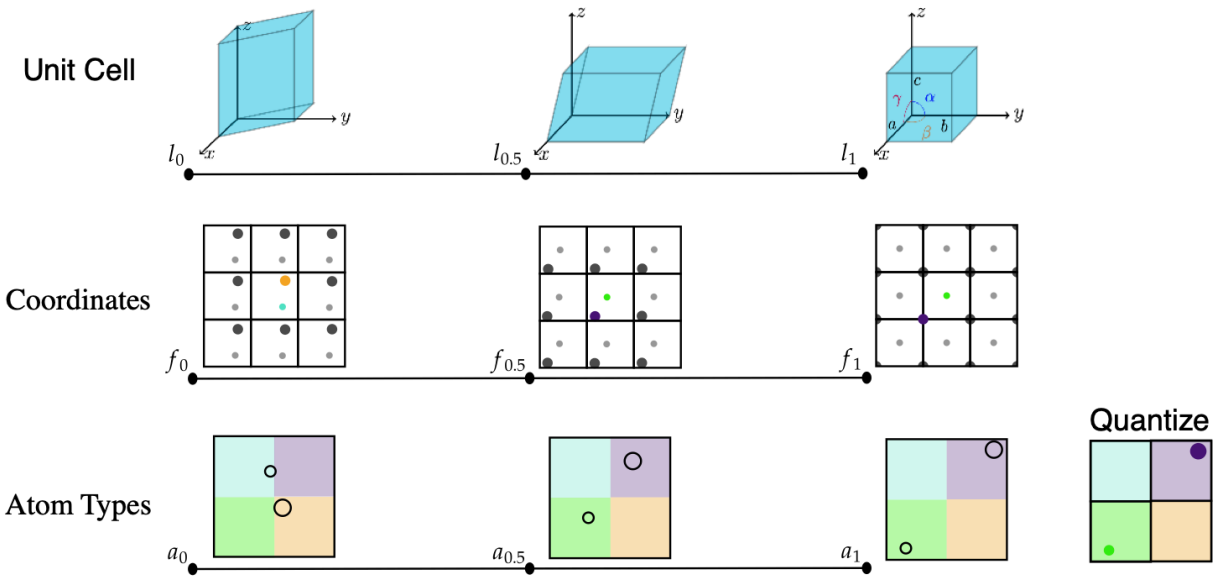
$$\boldsymbol{x}_1 = \boldsymbol{x}_0 + \int_0^1 \boldsymbol{v_\theta}(\boldsymbol{x}_t, t)\, dt$$

- Exactly learning $\boldsymbol{v_\theta}$ requires *full knowledge* of prior $p(\boldsymbol{x}_1)$
- Instead learn $\boldsymbol{u}_t(\boldsymbol{x}_1, \boldsymbol{x}_0)$ with linear paths between couples $(\boldsymbol{x}_0, \boldsymbol{x}_1)$, $\boldsymbol{x}_t = (1-t)\boldsymbol{x}_0 + t\boldsymbol{x}_1$



$$\implies \frac{d\boldsymbol{x}_t}{dt} = \boldsymbol{x}_1 - \boldsymbol{x}_0 \equiv \boldsymbol{u}_t$$

- Learn $\boldsymbol{A}, \boldsymbol{F}$, and $\boldsymbol{L}$ separately with deterministic paths, subject to their symmetry constraints



- $\boldsymbol{L}$: Unit cell initialized from physically-plausible cell parameters and flow pushes towards target
- $\boldsymbol{F}$: fractional coordinates are learned w.r.t crystal periodicity
- $\boldsymbol{A}$: represent atomic types in continuous embedding space and quantize results as a single type at end

Stochastic Interpolants Generally [5]

A stochastic interpolant learns a general mapping $\mathcal{M}_0 \to \mathcal{M}_1$ by

$$\boldsymbol{x}_t = I(t, \boldsymbol{x}_0, \boldsymbol{x}_1) + \gamma(t)\boldsymbol{z}$$

where

- $\boldsymbol{z}$ is the noise variable, typically $\mathcal{N}(0, \mathbb{1})$
- $\gamma(t)$ is the noise scheduler
- $I(t, \boldsymbol{x}_0, \boldsymbol{x}_1)$ is the interpolant satisfying $I(t = 0) = \boldsymbol{x}_0$ and $I(t = 1) = \boldsymbol{x}_1$

The neural network $\boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ then learns the conditional velocity field

$$\dot{\boldsymbol{x}}_t = \dot{I}(t, \boldsymbol{x}_0, \boldsymbol{x}_1) + \dot{\gamma}(t)\boldsymbol{z}$$

with

$$\mathcal{L} = \mathbb{E}_{t, \boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{z}}\left[\left\| \boldsymbol{b}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) - \dot{\boldsymbol{x}}_t \right\|^2\right]$$

Flow Matching as a Stochastic Interpolant

First assume the interpolant is linear in $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$:

$$\boldsymbol{x}_t = \alpha(t)\boldsymbol{x}_0 + \beta(t)\boldsymbol{x}_1 + \gamma(t)\boldsymbol{z}$$

Let $\alpha(t) = 1 - t, \beta(t) = t$, and $\gamma(t) = 0$:

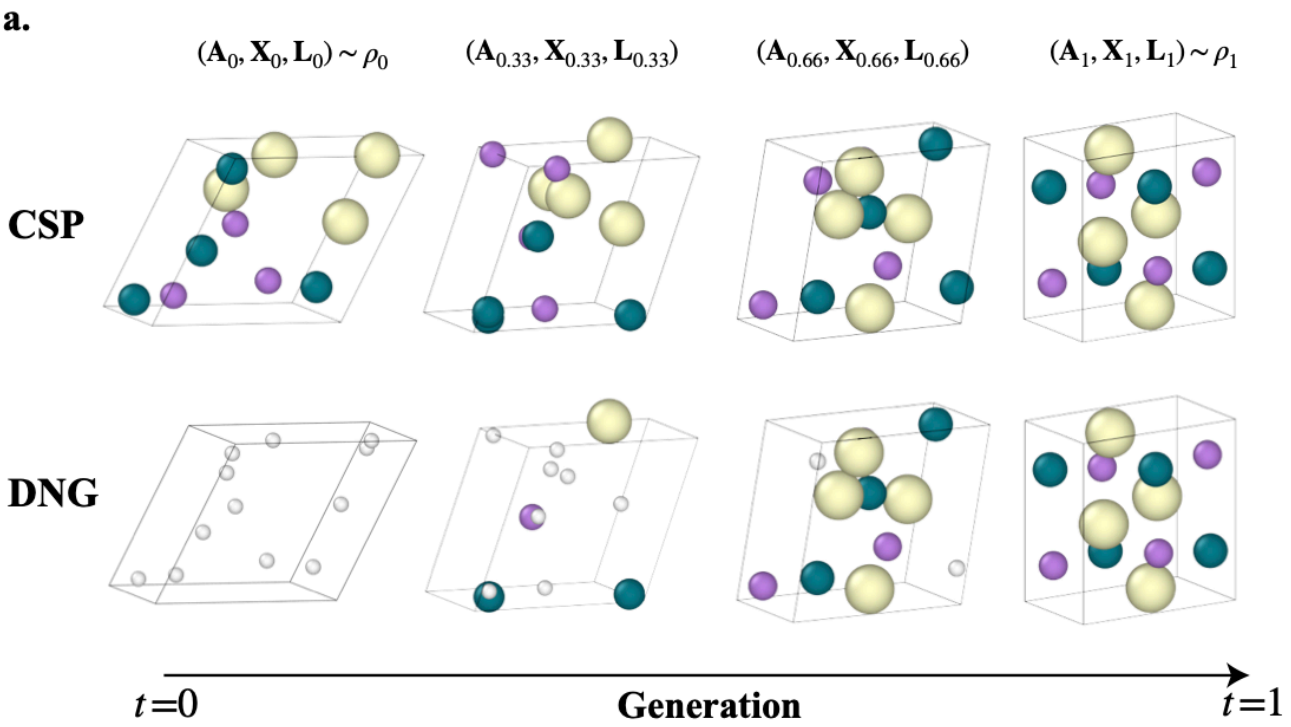
$$\boldsymbol{x}_t = (1 - t)\boldsymbol{x}_0 + t\boldsymbol{x}_1$$

...check that $I(0) = \mathbf{x}_0$ and $I(1) = \mathbf{x}_1$...

$$\implies \dot{\boldsymbol{x}}_t = \boldsymbol{x}_1 - \boldsymbol{x}_0 \equiv \boldsymbol{u}_t,$$

which is the same as the conditional flow matching objective

Stochastic Interpolants for Materials Generation: **OMatG** [6]

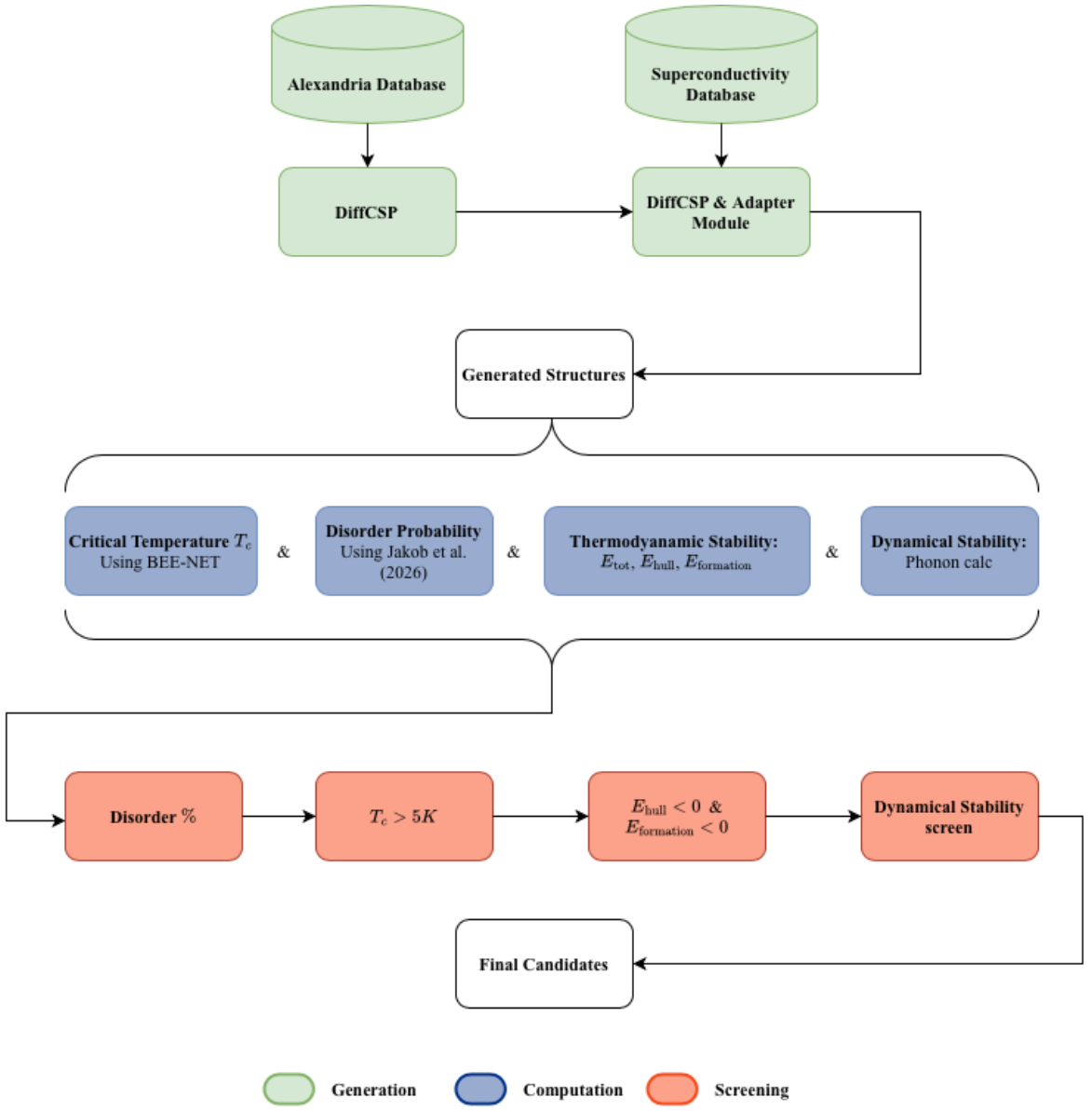


- Paths $\mathcal{M}_0 \longleftrightarrow \mathcal{M}_1$ are no longer fixed
- Optimal paths now can differ for $\boldsymbol{A}, \boldsymbol{F}, \boldsymbol{L}$

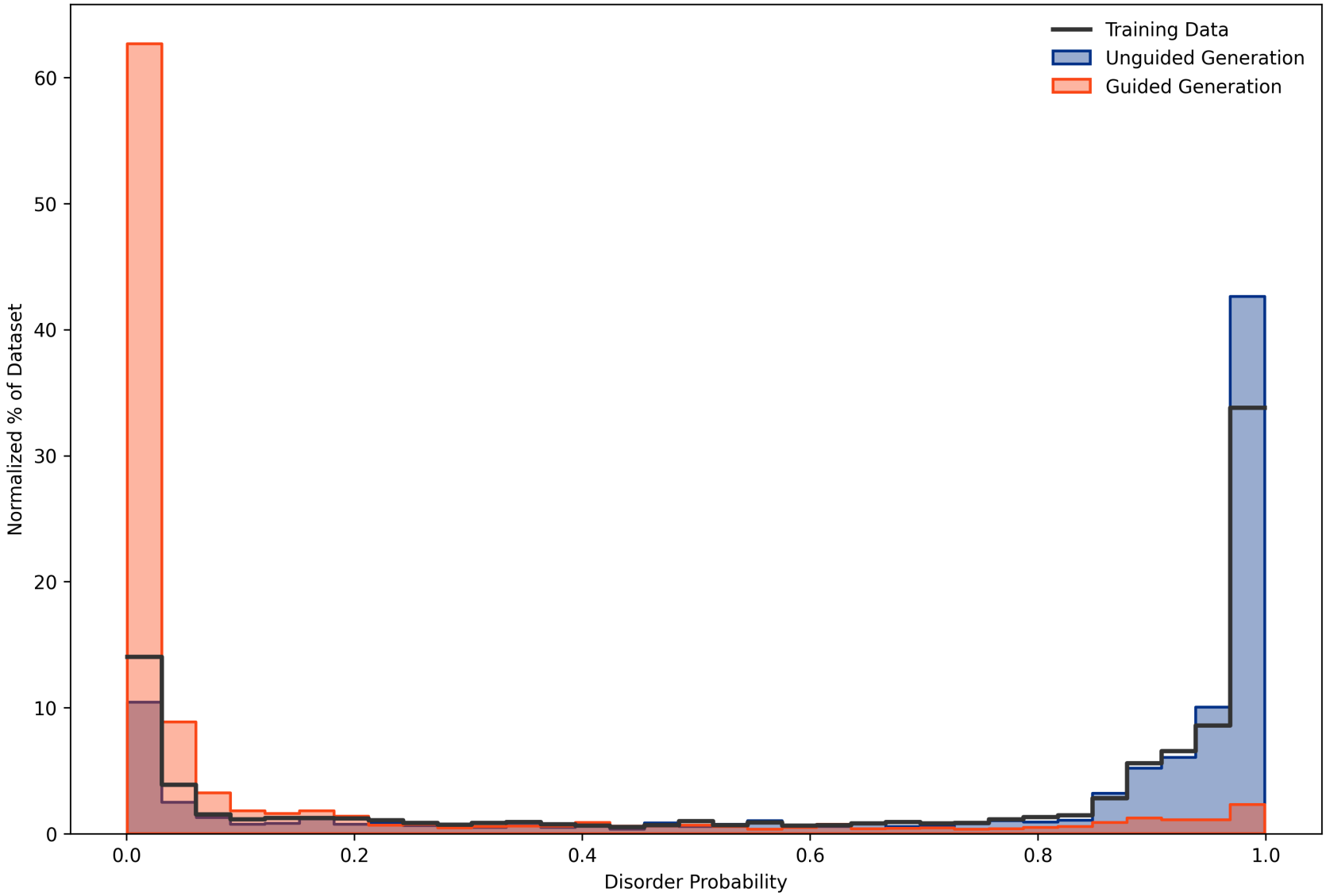
Framework Comparison

	DiffCSP	**FlowMM**	**OMatG**
Path type	Stochastic (SDE)	Deterministic (ODE)	Tunable
Lattice prior	Gaussian	Physically-informed	Physically-informed
Integration steps	$\sim$1000	$\sim$50	Tunable

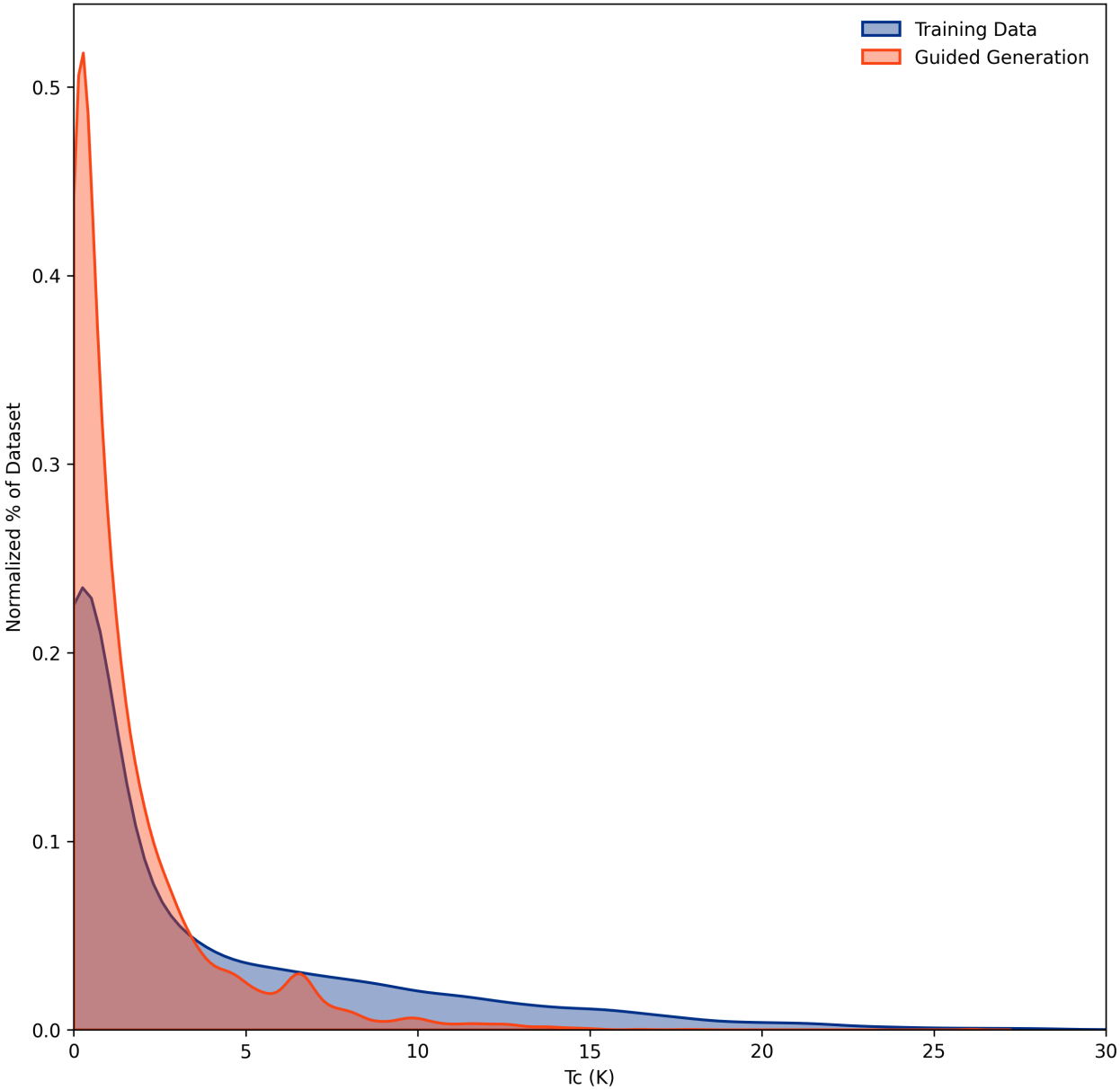
My Current Workflow



Learning Ordered Superconductors



Predicted T_c of Guided Data



Next Steps

Can learn low-disorder superconductors, but most are $T_c < 5$ K

- Swap foundation model DiffCSP $\rightarrow$ OMatG? (OMatG has reinforcement learning built-in)
- Add a second adapter module for T_c?

How well can pipelines be *expected* to extrapolate to high T_c?

Sources

[1] Jonathan Ho, Ajay Jain, and Pieter Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. arXiv: 2006.11239 [cs.LG]. URL: https://arxiv.org/abs/2006.11239.

[2] Rui Jiao et al. *Crystal Structure Prediction by Joint Equivariant Diffusion*. 2024. arXiv: 2309.04475 [cond-mat.mtrl-sci]. URL: https://arxiv.org/abs/2309.04475.

[3] Yaron Lipman et al. *Flow Matching for Generative Modeling*. 2023. arXiv: 2210.02747 [cs.LG]. URL: https://arxiv.org/abs/2210.02747.

[4] Benjamin Kurt Miller et al. *FlowMM: Generating Materials with Riemannian Flow Matching*. 2024. arXiv: 2406.04713 [cs.LG]. URL: https://arxiv.org/abs/2406.04713.

[5] Michael S. Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. *Stochastic Interpolants: A Unifying Framework for Flows and Diffusions*. 2025. arXiv: 2303.08797 [cs.LG]. URL: https://arxiv.org/abs/2303.08797.

[6] Philipp Hoellmer et al. *Open Materials Generation with Stochastic Interpolants*. 2025. arXiv: 2502.02582 [cs.LG]. URL: https://arxiv.org/abs/2502.02582.