

Continuous Generative Models

Andrew Valentini

University of Florida

July 2026

General Goal:

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

Specific Goal:

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

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

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

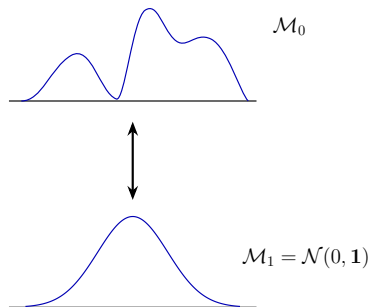
- $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_N] \in \mathbb{R}^{3 \times 118}$ are one-hot encodings of atomic type at each site
- $\mathbf{F} = [\mathbf{f}_1, \mathbf{f}_2, \dots, \mathbf{f}_N] \in [0, 1)^{3 \times N}$ are fractional coordinates of the atoms
 - $p(\mathbf{F}_0)$ must satisfy translation symmetry invariance due to crystal periodicity
- $\mathbf{L} = [\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \boldsymbol{\ell}_3] \in \mathbb{R}^{3 \times 3}$ are the basis vectors of unit cell
 - $p(\mathbf{L}_0)$ 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 \mathcal{M}

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

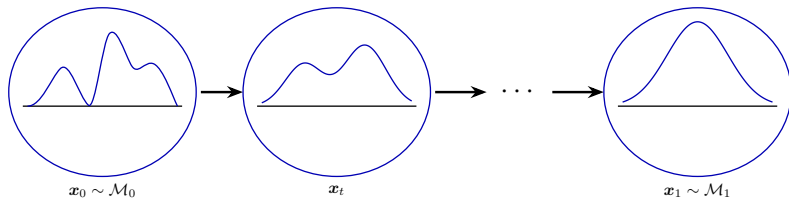
- 1 Let a neural net map $\mathcal{M}_0 \rightarrow \mathcal{N}(0, \mathbb{1})$
- 2 Have it learn the reverse process

Three of such models:

- Diffusion
- Flow Matching
- Stochastic Interpolants



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



- For a random step $t \sim U(0, 1)$, sample $x_0 \sim \mathcal{M}_0$ by

$$x_{t+1} = a_t x_t + b_t \epsilon_t$$

With $\mathbf{x}_{t+1} = a_t\mathbf{x}_t + b_t\boldsymbol{\epsilon}_t$, determine a_t & b_t by forcing $\text{Var}[\mathbf{x}_t] = 1$

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

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

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

where α_t is a **noise scheduler**, defined such that $\alpha_1 = 0$ to ensure $\mathbf{x}_t \rightarrow \mathcal{N}(0, \mathbb{1})$ as $t \rightarrow 1$

With the forward diffusion process

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

let a neural network learn to predict the cumulative noise $\boldsymbol{\epsilon}$ added to a sample \mathbf{x}_0 at a specific timestep t :

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

Then invert the diffusion process ($\mathbf{x}_{t+1} \rightarrow \mathbf{x}_t$) 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 $q(\mathbf{x}_{t+1} | \mathbf{x}_t) = \mathcal{N}(z; \mu, \sigma)$:

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

To sample from $\mathcal{M}(\mathbf{A}, \mathbf{F}, \mathbf{L})$ 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 $\mathbf{X} = [\mathbf{A}, \mathbf{L}]$:

$$q(\mathbf{X}_t, \mathbf{X}_0) = \mathcal{N}(\mathbf{X}_t | \sqrt{\alpha_t}\mathbf{X}_0, (1 - \alpha_t)\mathbb{1})$$

$$\& \mathcal{L}_{\mathbf{X}} = \mathbb{E}_{\epsilon_{\mathbf{X}}, t} [\|\epsilon_{\mathbf{X}} - \hat{\epsilon}_{\mathbf{X}}\|^2]$$

where $E(3)$ symmetry for \mathbf{L} is enforced in the transitions $\mathbf{x}_t \rightarrow \mathbf{x}_{t+1}$

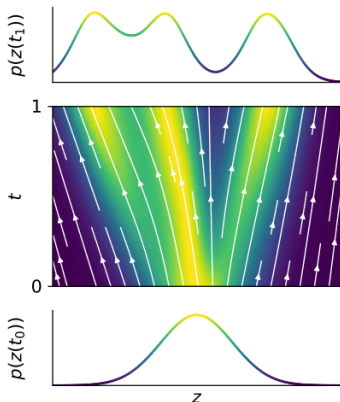
- Diffusion for \mathbf{F} is more complicated to enforce translational symmetry

Idea: Replace diffusion's stochastic process by deterministic mapping \mathbf{v}_θ (many fewer steps required)

- Learned \mathbf{v}_θ then generates new samples by

$$\mathbf{x}_1 = \mathbf{x}_0 + \int_0^1 \mathbf{v}_\theta(\mathbf{x}_t, t) dt$$

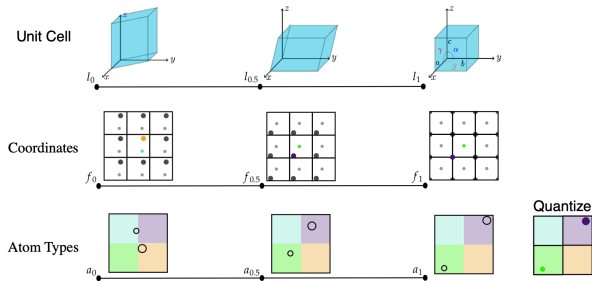
- Exactly learning \mathbf{v}_θ requires *full knowledge* of prior $p(\mathbf{x}_1)$
- Instead learn $\mathbf{u}_t(\mathbf{x}_1, \mathbf{x}_0)$ with linear paths between couples $(\mathbf{x}_0, \mathbf{x}_1)$, $\mathbf{x}_t = (1 - t)\mathbf{x}_0 + t\mathbf{x}_1$



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

Flow Matching for Materials Generation: FlowMM [4]

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



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

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

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

where

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

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

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

with

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

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

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

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

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

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

$$\implies \dot{\mathbf{x}}_t = \mathbf{x}_1 - \mathbf{x}_0,$$

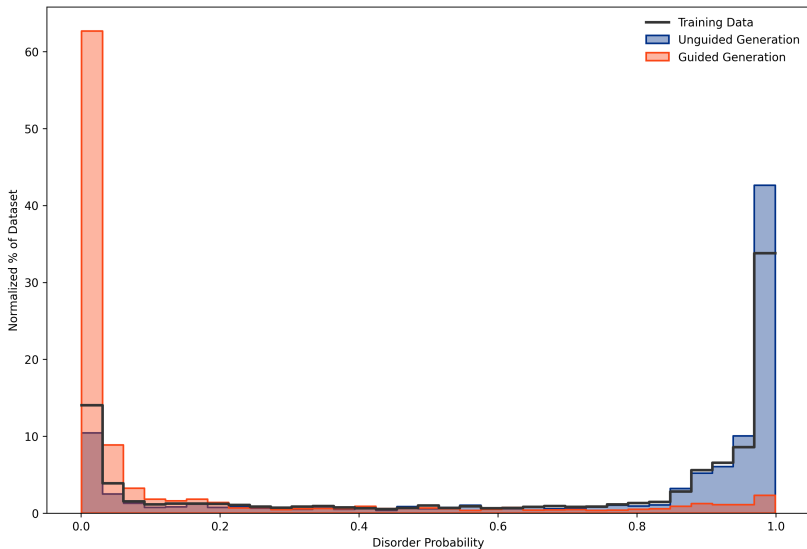
which is the same as the conditional flow matching objective

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

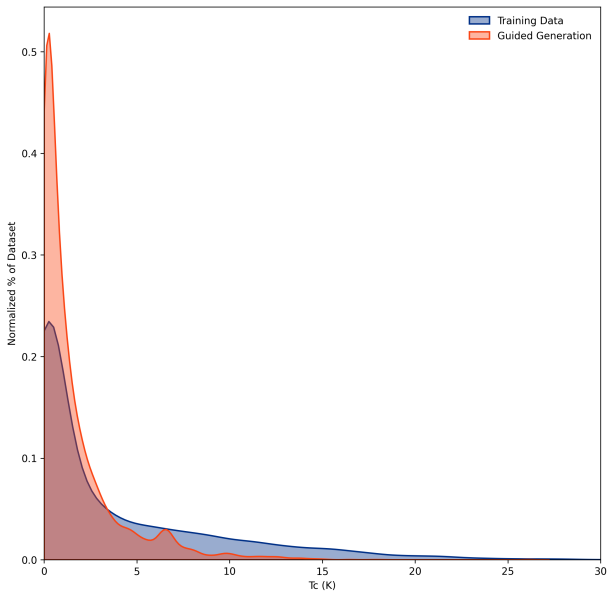
My Current Workflow



Learning Ordered Superconductors



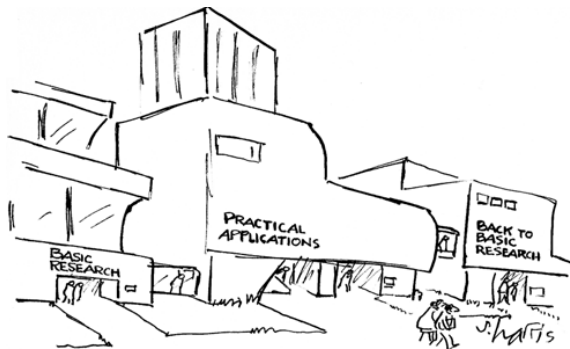
Predicted T_c of Guided Data



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 ?

- [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>.