Simulate Time-integrated Coarse-grained Molecular Dynamics with Geometric Machine Learning

Authors

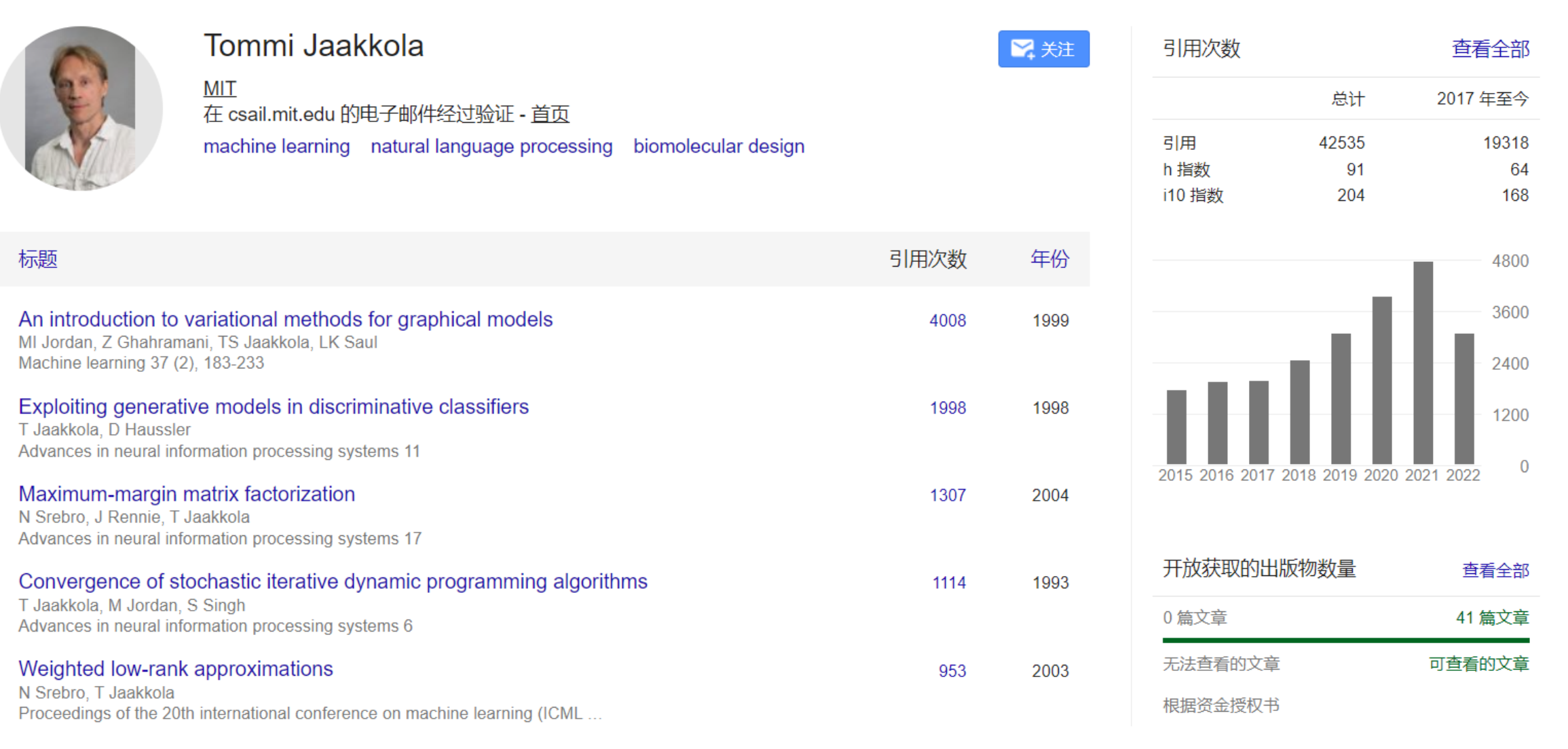
Simulate Time-integrated Coarse-grained Molecular Dynamics with Geometric Machine Learning

Xiang Fu[1*†], Tian Xie[1†], Nathan J. Rebello[2], Bradley D. Olsen[2] and Tommi Jaakkola[1*]

[1*]CSAIL, MIT, Cambridge, 02139, MA, United States.
[2]Department of Chemical Engineering, MIT, Cambridge, 02139, MA, United States.

*Corresponding author(s). E-mail(s): xiangfu@csail.mit.edu; tommi@csail.mit.edu;
Contributing authors: txie@csail.mit.edu; nrebello@mit.edu; bdolsen@mit.edu;
†Equal Contribution.

About the last author



Tommi Jaakkola

关注

MIT

在 csail.mit.edu 的电子邮件经过验证 - 首页

machine learning natural language processing biomolecular design

标题	引用次数	年份
An introduction to variational methods for graphical models MI Jordan, Z Ghahramani, TS Jaakkola, LK Saul Machine learning 37 (2), 183-233	4008	1999
Exploiting generative models in discriminative classifiers T Jaakkola, D Haussler Advances in neural information processing systems 11	1998	1998
Maximum-margin matrix factorization N Srebro, J Rennie, T Jaakkola Advances in neural information processing systems 17	1307	2004
Convergence of stochastic iterative dynamic programming algorithms T Jaakkola, M Jordan, S Singh Advances in neural information processing systems 6	1114	1993
Weighted low-rank approximations N Srebro, T Jaakkola Proceedings of the 20th international conference on machine learning (ICML ...	953	2003

引用次数 查看全部

	总计	2017 年至今
引用	42535	19318
h 指数	91	64
i10 指数	204	168

开放获取的出版物数量 查看全部

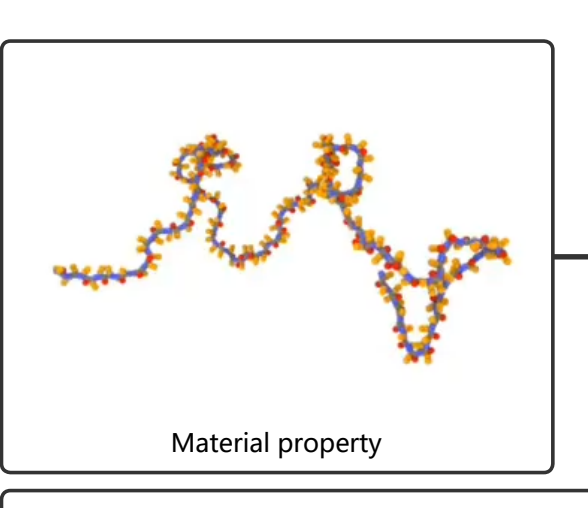
0 篇文章 41 篇文章

无法查看的文章 可查看的文章

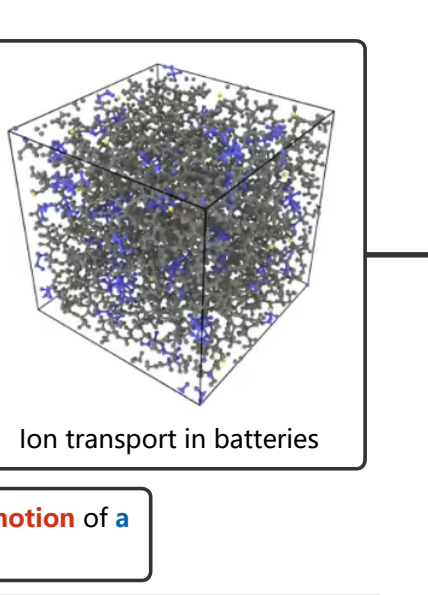
根据资金授权书

Background

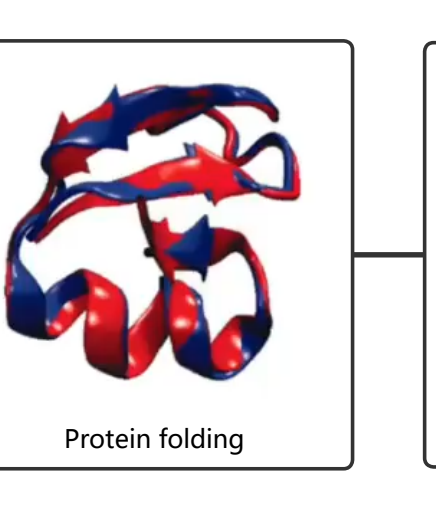
Molecular dynamic



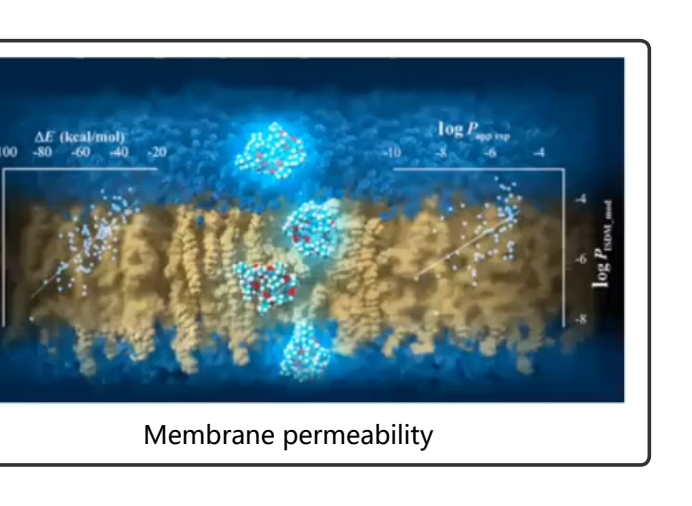
Material property



Ion transport in batteries



Protein folding



Membrane permeability

Molecular dynamic simulation: Simulation of the motion of a large number of atoms within a system.

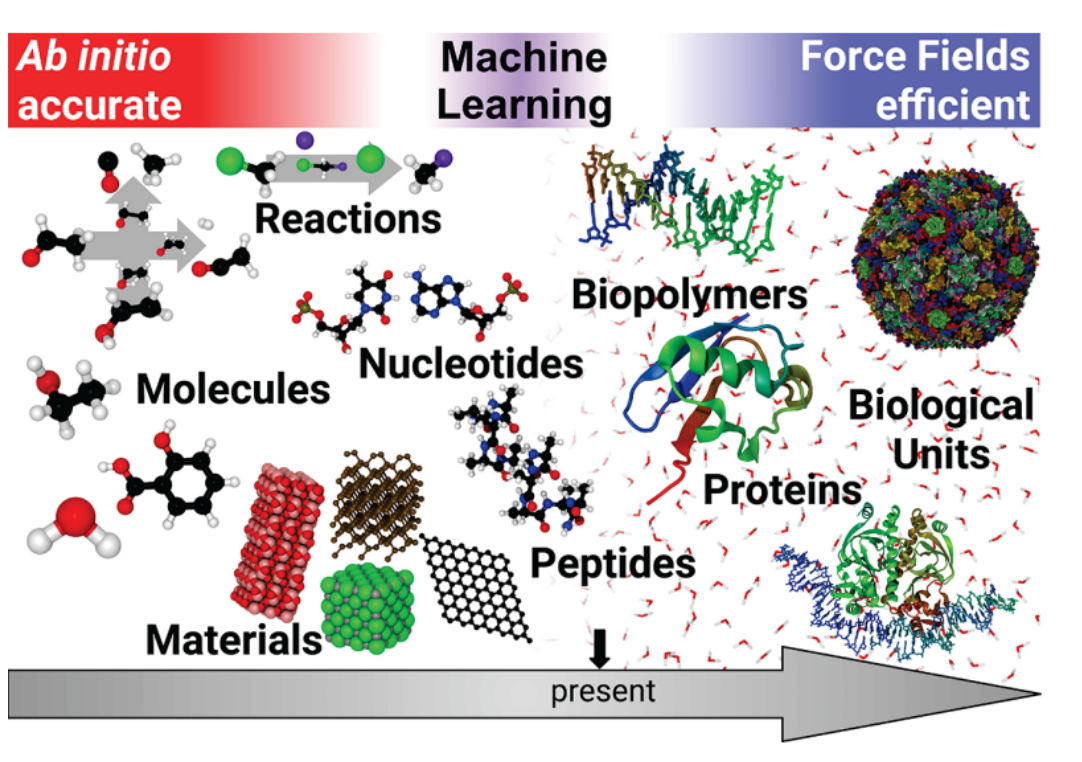
Motivation: fast molecular dynamics (MD) simulation

- System is complex: tens of thousands of atoms
- Long simulation: nanosecond to microsecond level
- Large chemical space to simulate

High computational cost or even infeasible

Related work

Machine learning force field method



Ab initio method: 是一种基于量子化学的方法。通过求解与系统内所有原子相关的薛定谔方程得到精确的原子之间作用力，进而模拟系统内的分子动力学-->很精确，但是很慢，在原子数较多时基本不可行

Force fields method: 是一种基于经典力学的方法。原子之间的相互作用建模成键合力和非键合力-->一般快，不是很精确

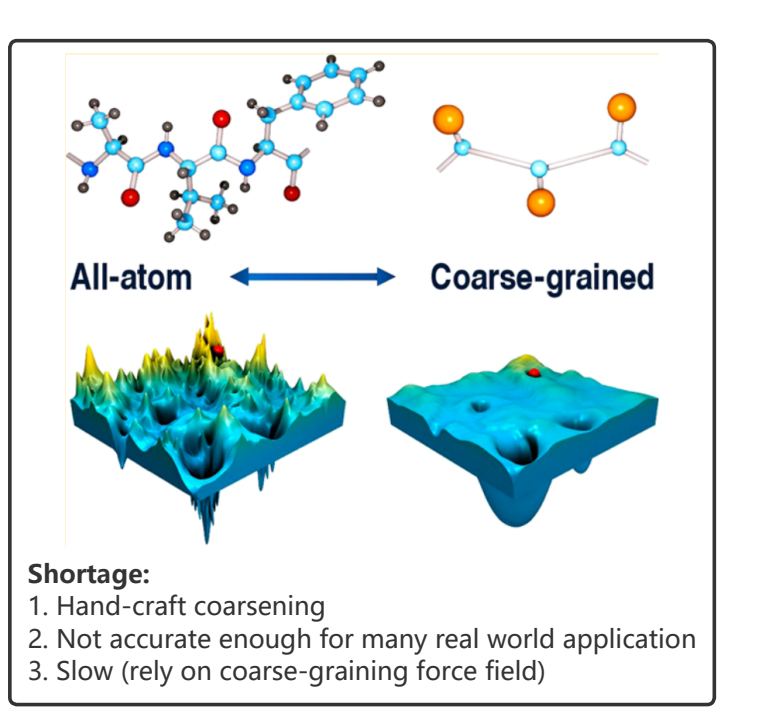
键合力：通过键合原子之间的与距离相关的简单函数来描述，或者用原子之间的角度和二面角来描述。

非键合力：通过计算所有原子两两之间库伦力来描述。

Machine learning force field: 是一种基于机器学习逼近量子化学计算精度的方式。通过supervise的方法，输入系统内的所有原子，输出原子之间的相互作用关系。-->一般快，比较精确

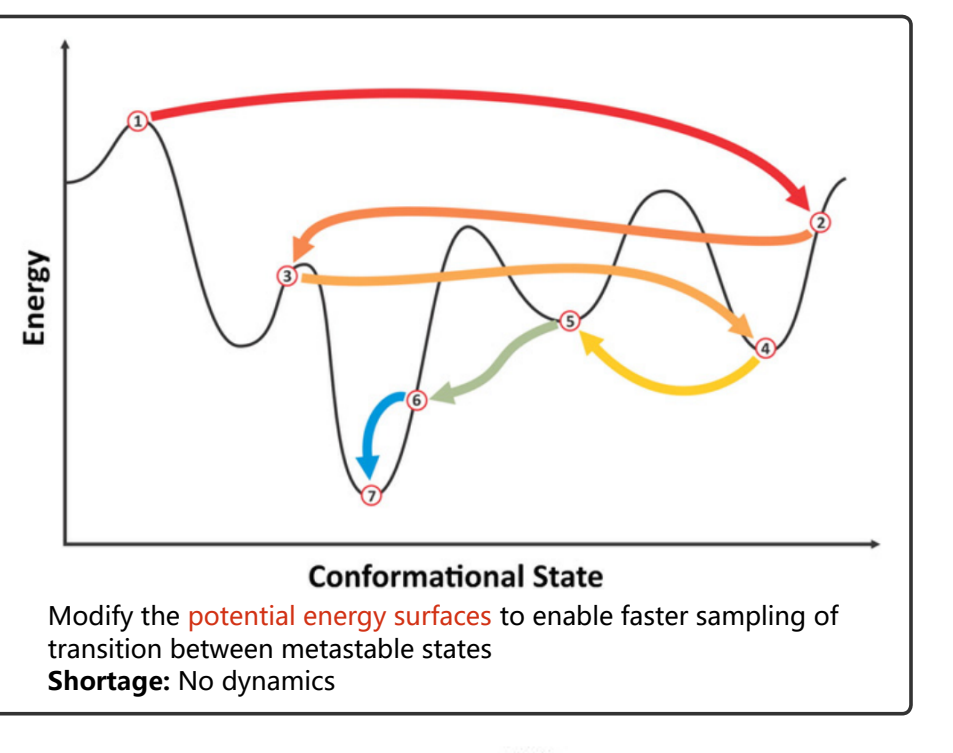
Coarse-graining model

Reduce system complexity



Shortage:
1. Hand-craft coarsening
2. Not accurate enough for many real world application
3. Slow (rely on coarse-graining force field)

Enhanced sampling method

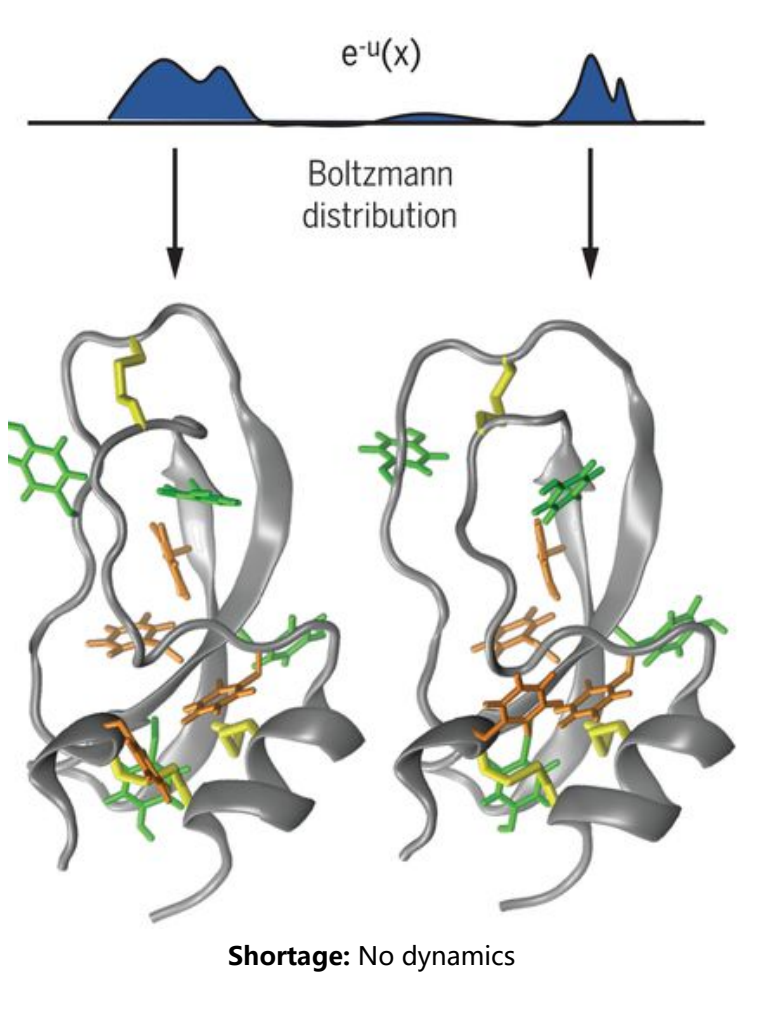


Conformational State

Modify the potential energy surfaces to enable faster sampling of transition between metastable states

Shortage: No dynamics

Machine learning generative modelling



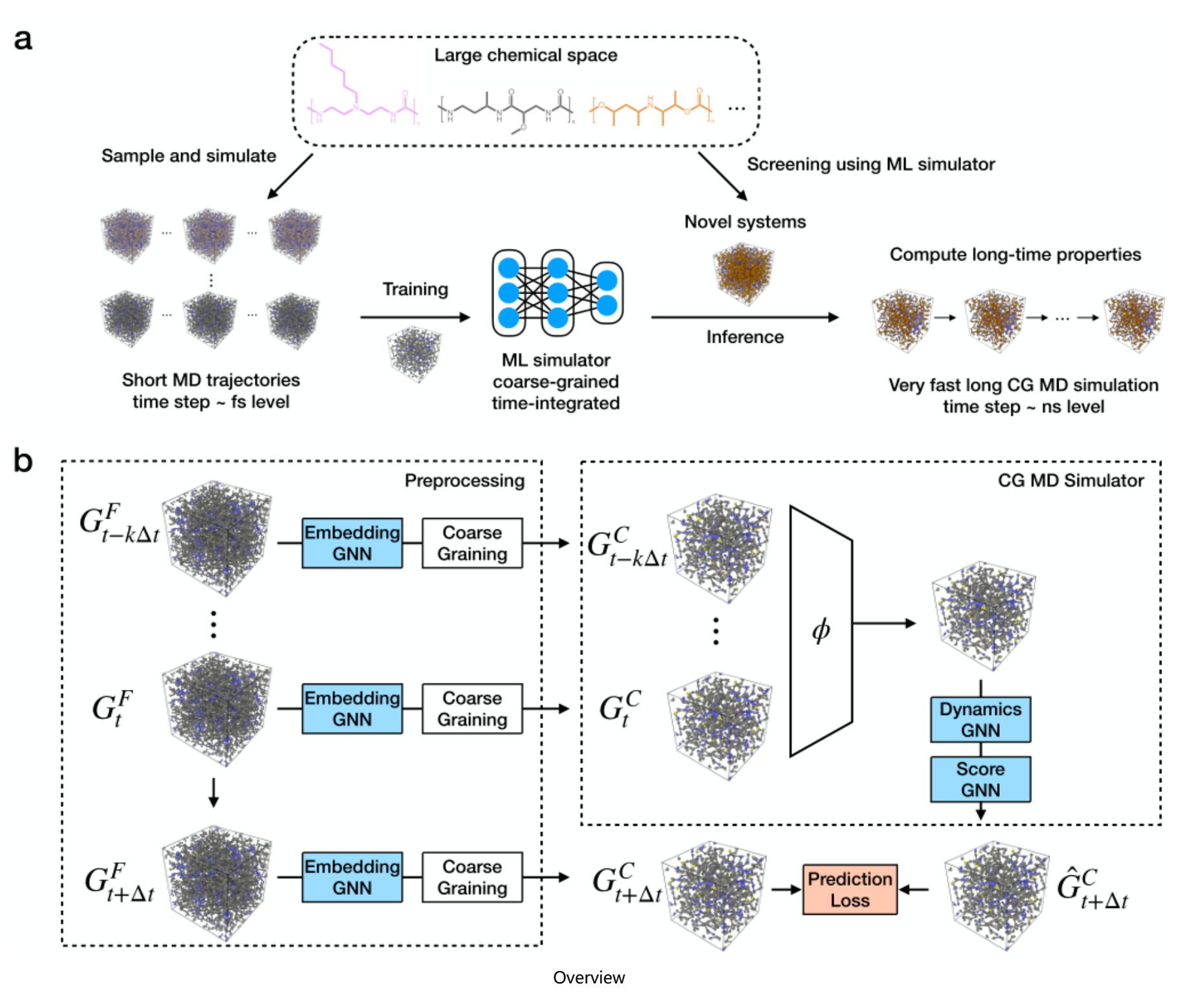
Shortage: No dynamics

Method

Solutions to the above shortage

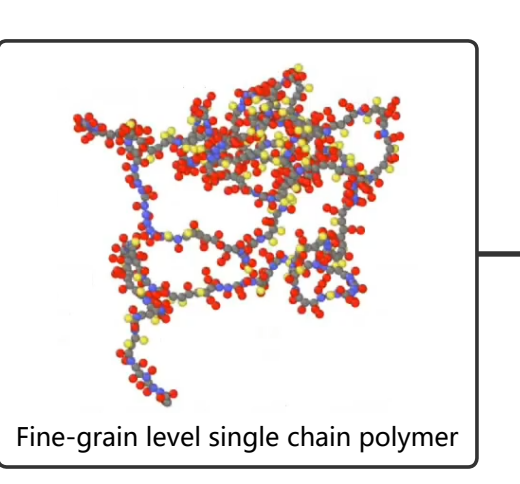
- System is complex: tens of thousands of atoms — Spatial coarse-graining
- Long simulation: nanosecond to microsecond level — Very large time-integration step at nanosecond-level without force computation
- Large chemical space to simulate — Learn from short MD trajectories and simulate long trajectories for novel unseen systems

Framework

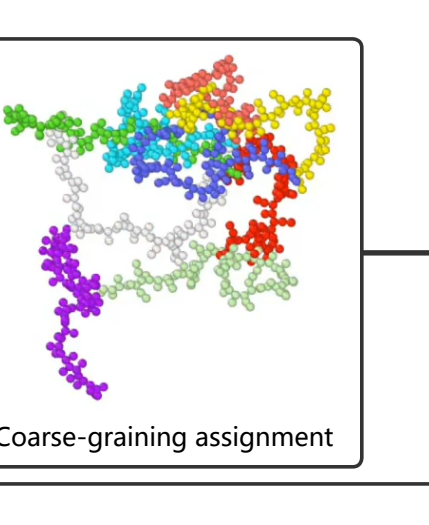


Overview

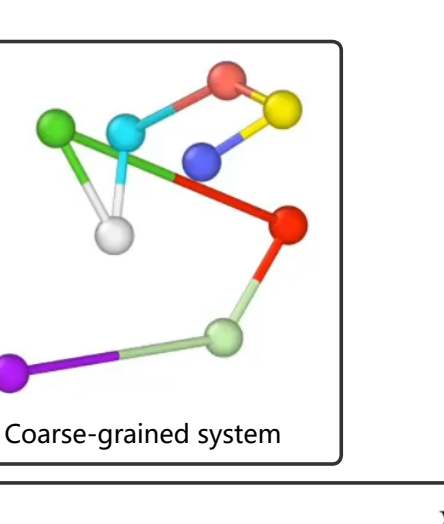
Preprocessing step



Fine-grain level single chain polymer



Coarse-graining assignment



Coarse-grained system

Embedding GNN

Graph processing with graph neural networks

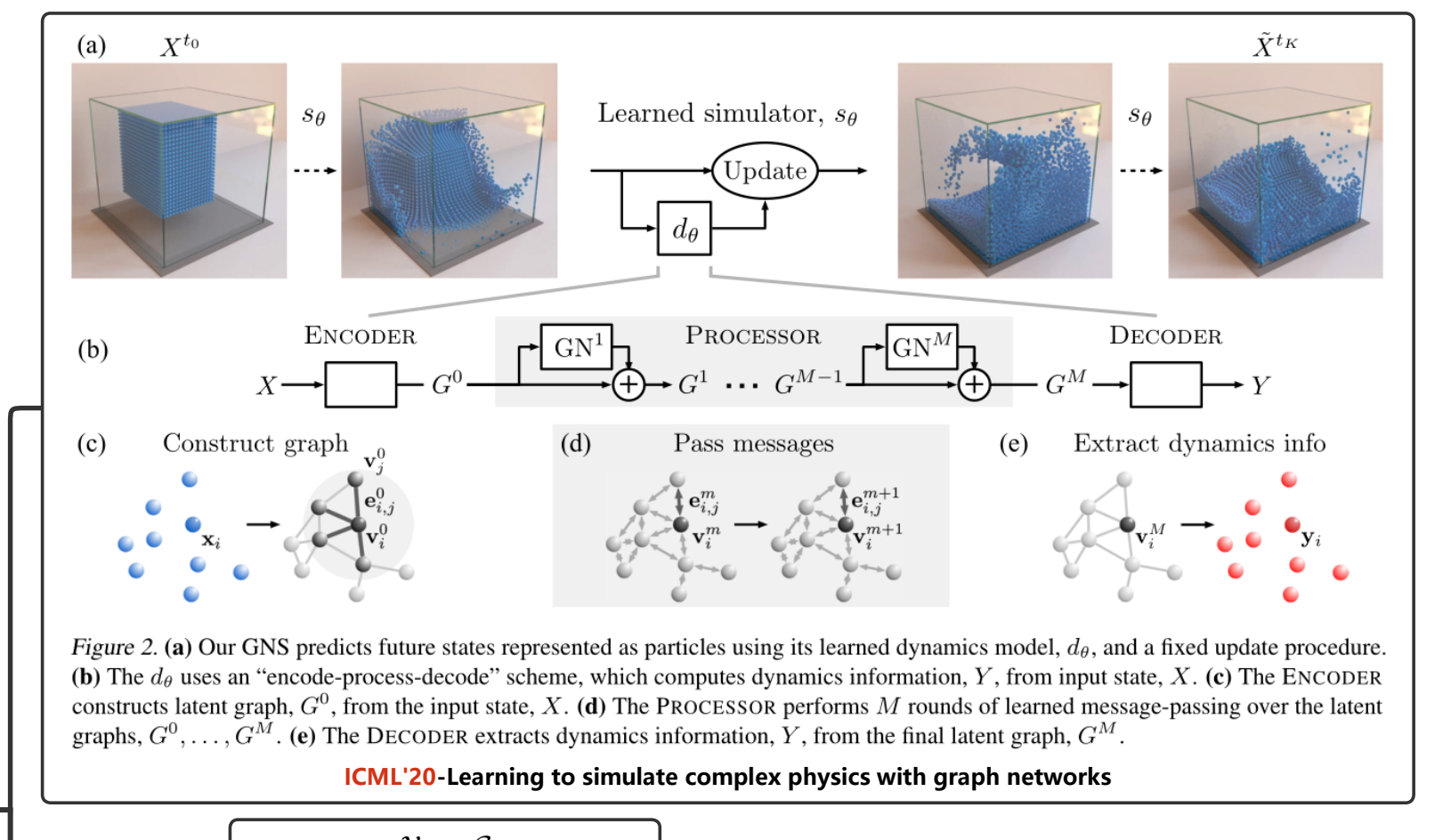


Figure 2. **(a)** Our GNS predicts future states represented as particles using its learned dynamics model, d_θ, and a fixed update procedure. **(b)** The d_θ uses an "encode-process-decode" scheme, which computes dynamics information, Y, from input state, X. **(c)** The ENCODER constructs latent graph, G^0, from the input state, X. **(d)** The PROCESSOR performs M rounds of learned message-passing over the latent graphs, $G^0, \ldots, G^M$. **(e)** The DECODER extracts dynamics information, Y, from the final latent graph, G^M.

ICML'20-Learning to simulate complex physics with graph networks

Encoder

- $\mathcal{X} \rightarrow \mathcal{G}$
  Objective: Learn a graph from give X
- $\mathbf{v}_i = \varepsilon^v(\mathbf{x}_i)$
  E^v is a **2-layers MLP** that is independently applied to each node
- $\mathbf{e}_{i,j} = \varepsilon^e(\mathbf{r}_{i,j})$
  E^e is a **2-layers MLP** that is independently applied to each edge
  - $r_{i,j}$ can be displacement between the positions of node i and node j

Processor

- $\mathbf{G} = (G^1, ..., G^M)$
  Objective: Generate a sequence of update latent graph
- $G^{m+1} = \mathrm{GN}^{m+1}(G^m)$
  GN is a **7 layers MPNN**

Decoder

- $\mathcal{G} \rightarrow \mathcal{Y}$
- $\mathbf{y}_i = \delta^v(\mathbf{v}_i^M)$
  δ^v is a **2-layers MLP** that is independently applied to each node

Representing MD trajectories as time series of graphs

Time variant graph: MD simulation trajectory is represented as a time series of fine-level graphs $\{G_t^F\}$. Each fine-level graph is represented as $G_t^F = (V_t^F, E_t^F)$

- $\boldsymbol{v}^F_{i,t} \in V^F_t$
  atom information: invariant and variant feature
- $\boldsymbol{e}^F_{i,j} \in E^F$
  chemical bond

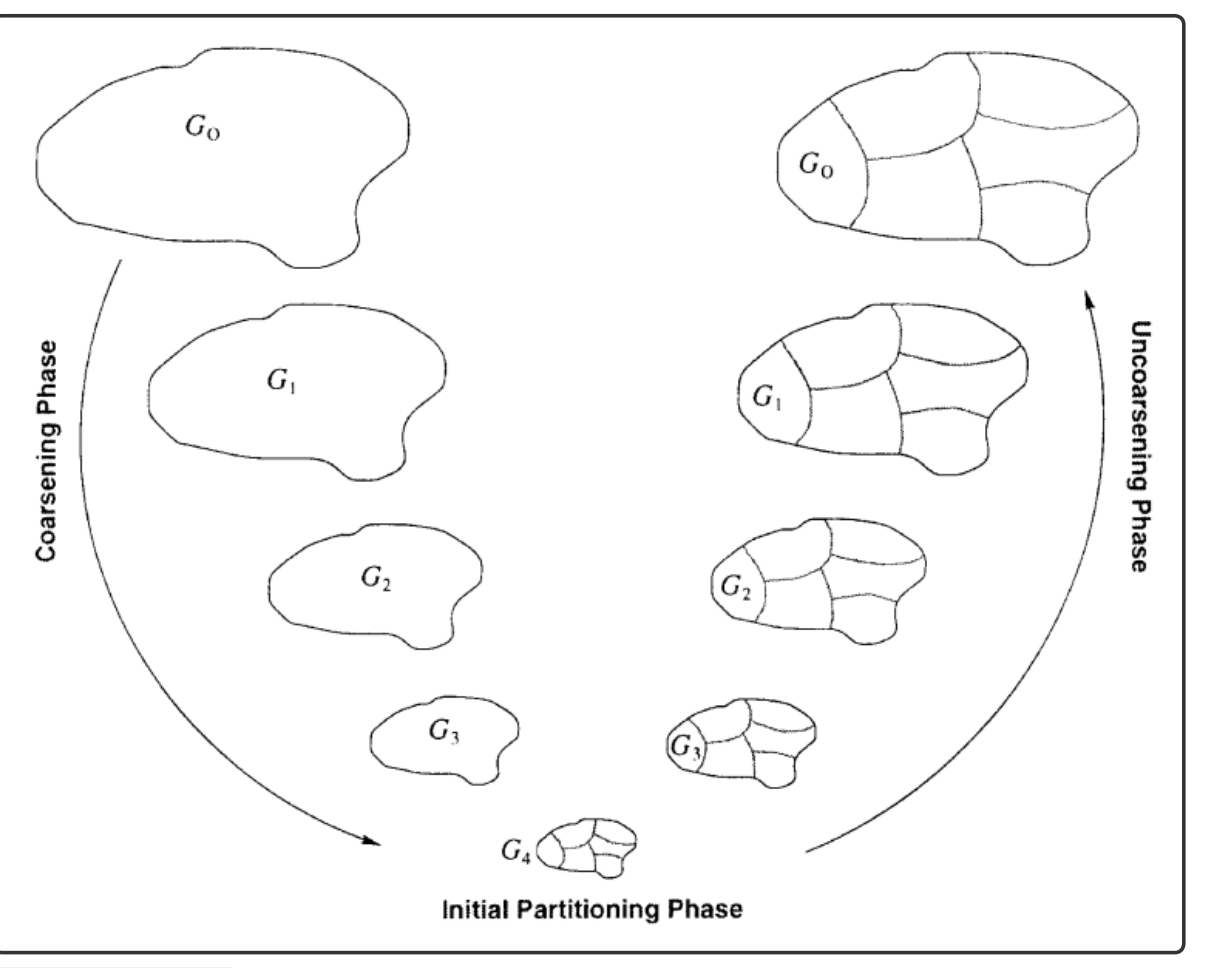
Learning CG-bead type embeddings with embedding GNN

- **Objective:** Learn the node embeddings which contains **no positional information**
- $G^F = (V^F, E^F)$
  Input: Time invariant graph that only contains atom type, atom weights, bond types information
  - $\boldsymbol{v}_i^F = [\boldsymbol{a}_i^F, w_i^F]$
    learnable atom type embedding: vector
    weight: scale
  - $\boldsymbol{e}^F_{i,j} = [\boldsymbol{a}_i^F + \boldsymbol{a}_j^F + \boldsymbol{a}_{i,j}^F]$
    learnable bond type embedding: vector
- **Output:** node type embedding c_i^F for all $v_i^F \in V^F$

Coarse-graining

Graph clustering CG model

METIS



Construct CG graph states

- **Objectives**: grouping atoms in the same group into a CG-bead
- **Fine-level node feature:** $\mathbf{v}_{i,t}^F = [\mathbf{c}_i^F, w_i^F, \mathbf{x}_{i,t}^F]$
- **Create CG-bead:** $\mathbf{v}_{m,t}^C = [\mathbf{c}_m^C, w_m^C, \mathbf{x}_{m,t}^C]$

$$\boldsymbol{c}_m^C = \underset{C_m}{\mathtt{mean}}(\boldsymbol{c}_i^F) \equiv \frac{\sum_{i\in C_m} \boldsymbol{c}_i^F}{|C_m|}$$

$$w_m^C = \underset{C_m}{\mathtt{sum}}(w_i^F) \equiv \sum_{i\in C_m} w_i^F$$

$$\boldsymbol{x}_{m,t}^C = \underset{C_m}{\mathtt{CoM}}(\boldsymbol{q}_{i,t}^F, w_i^F) \equiv \frac{\sum_{i\in C_m} w_i^F \boldsymbol{x}_{i,t}^F}{\sum_{i\in C_m} w_i^F}$$

- $e^C_{m,n,t} \in E^C_t \Longleftarrow \exists i \in C_m, j \in C_n,$ such that $e^F_{i,j} \in E^F$
  Create CG-bond: if there exists a chemical bond between a pair of atoms in group C_m and group C_n
- $e^C_{m,n,t} \in E^C_t \Longleftarrow \|\boldsymbol{x}^C_{m,t} - \boldsymbol{x}^C_{n,t}\| < r$
  Create radius cut-off edge

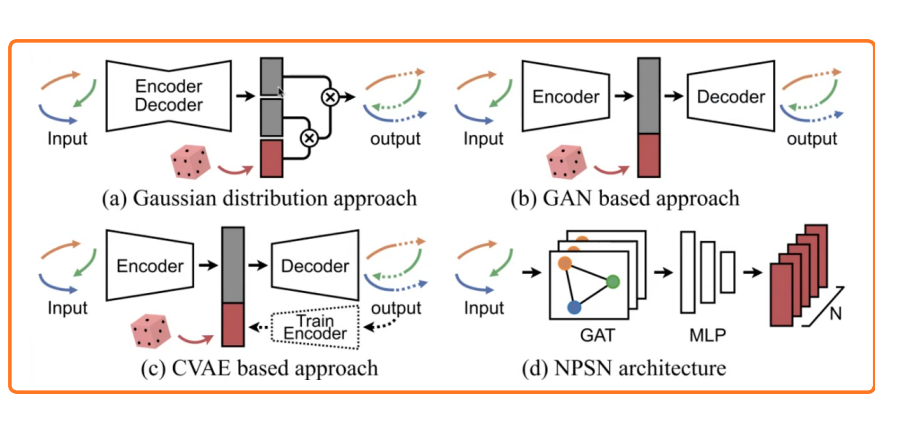
Learning CG MD with Dynamics GNN

Input

- $\boldsymbol{v}^C_{m,t} = [\boldsymbol{c}^C_m, w^C_m, \{\dot{\boldsymbol{x}}^C_{m,t-i\Delta t}\}_{i=0}^{k-1}]$
  Node feature
- $e^C_{m,n,t} = [\boldsymbol{x}^C_{m,t} - \boldsymbol{x}^C_{n,t}, \|\boldsymbol{x}^C_{m,t} - \boldsymbol{x}^C_{n,t}\|, \boldsymbol{c}^C_{m,n}]$
  Edge feature, $c^C_{m,n}$ indicate whether $e^C_{m,n,t}$ is a CG-bond or is constructed through radius cut-off

Dynamics GNN

- Encoder — MPNN
- Decoder — MLP on Node to learn the acceleration
- $\mathrm{GN}_D(\phi(\{G^C_{t-i\Delta t}\}_{i=0}^k)) = \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t^2) = \{\mathcal{N}(\boldsymbol{\mu}_{m,t}, \boldsymbol{\sigma}^2_{m,t}) : \boldsymbol{v}^C_{m,t} \in V_t^C\}$
  Finally return 3-dimensional Gaussian distribution of acceleration for each coarsen-grained node
- Loss — $L_{\mathrm{dyn}} = -\log \mathcal{N}(\ddot{\boldsymbol{x}}_t^C | \boldsymbol{\mu}_t, \boldsymbol{\sigma}_t^2)$



(a) Gaussian distribution approach
(b) GAN based approach
(c) CVAE based approach
(d) NPSN architecture

Predict position

$$\hat{\dot{\boldsymbol{x}}}_{t+\Delta t} = \dot{\boldsymbol{x}}_t + \hat{\ddot{\boldsymbol{x}}}_t \Delta t$$

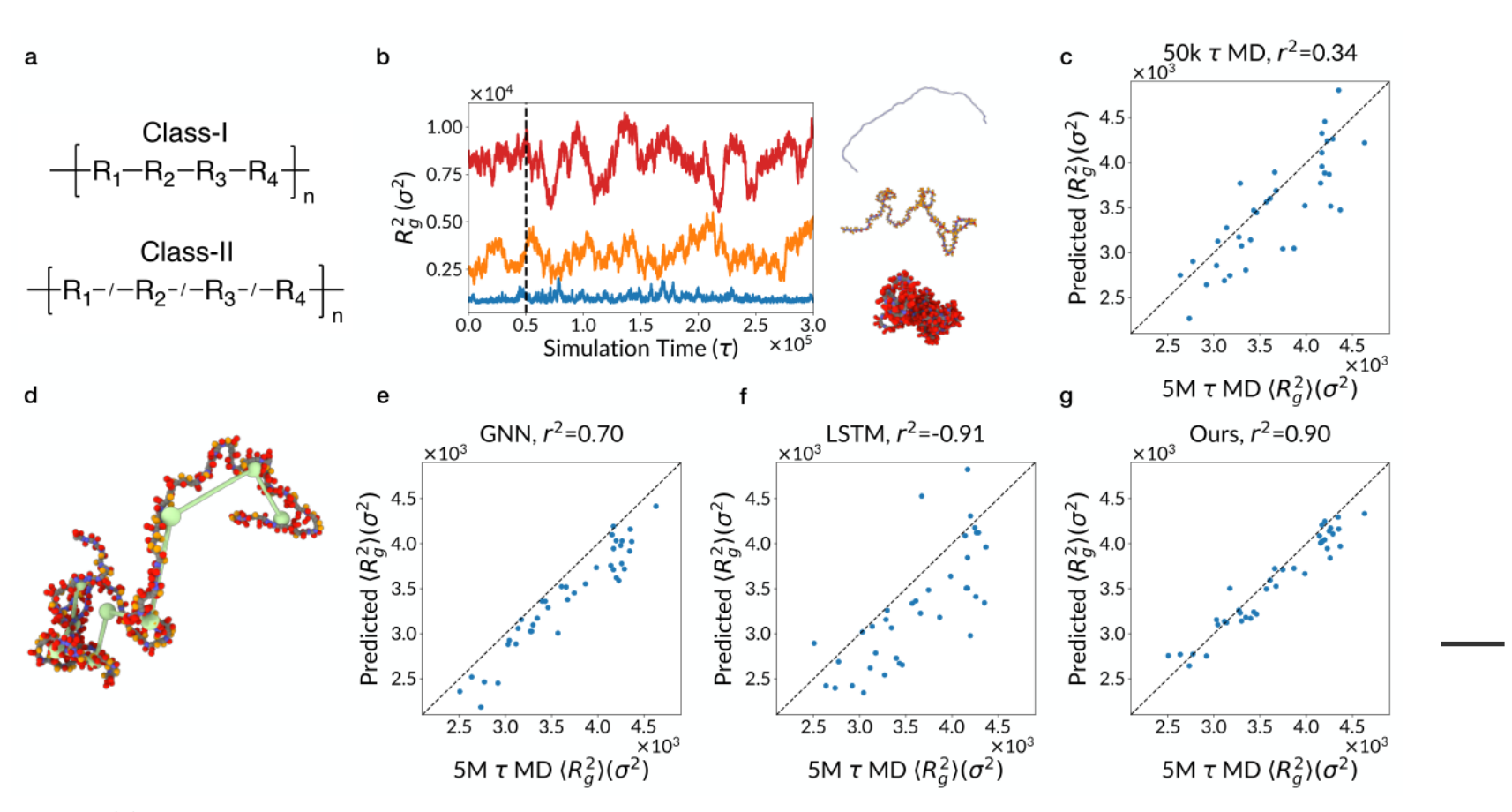
$$\hat{\boldsymbol{x}}_{t+\Delta t} = \boldsymbol{x}_t + \hat{\dot{\boldsymbol{x}}}_{t+\Delta t} \Delta t$$

Predict the position using Euler integration

Learning to refine CGMD predictions with Score GNN

- **Noise Conditional Score Network** (NCSN): **NeurIPS'19** - Generative Modeling by Estimating Gradients of the Data Distribution
- **Goal of NCSN here:** Refine the results from **Dynamics GNN**
- **Langevin dynamics** — $\tilde{\mathbf{x}}_t = \tilde{\mathbf{x}}_{t-1} + \frac{\epsilon}{2}\nabla_{\mathbf{x}} \log p(\tilde{\mathbf{x}}_{t-1}) + \sqrt{\epsilon}\, \mathbf{z}_t,$
- **Goal of NCSN**
  - $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_{\mathrm{data}}(\mathbf{x})$
    Train a score network
    - $\min_\theta \frac{1}{2}\mathbb{E}_{p_{\mathrm{data}}(\mathrm{x})}[\|\mathrm{s}_\theta(\mathrm{x}) - \nabla_{\mathrm{x}} \log p_{\mathrm{data}}(\mathrm{x})\|_2^2],$
      Loss function
  - $\min_\theta \frac{1}{2}\mathbb{E}_{q_\sigma(\tilde{\mathrm{x}}|\mathrm{x})p_{\mathrm{data}}(\mathrm{x})}[\|\mathrm{s}_\theta(\tilde{\mathrm{x}}) - \nabla_{\mathrm{x}} \log q_\sigma(\tilde{\mathrm{x}}|\mathrm{x})\|_2^2]$
    - $q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}} \mid \mathbf{x}, \sigma^2 I)$
    - $\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) = -(\tilde{\mathbf{x}}-\mathbf{x})/\sigma^2$
  - $\ell(\boldsymbol{\theta}; \sigma) \triangleq \frac{1}{2}\mathbb{E}_{p_{\mathrm{data}}(\mathbf{x})}\mathbb{E}_{\tilde{\mathbf{x}}\sim\mathcal{N}(\mathbf{x},\sigma^2 I)}\left[\left\|\mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, \sigma) + \frac{\tilde{\mathbf{x}} - \mathbf{x}}{\sigma^2}\right\|_2^2\right]$

Result



$$R_g^2 = \left(\frac{\sum_{i\in V} m_i d_i^2}{\sum_{i\in V} m_i}\right)$$

Fig. 2 (a) Class-I polymers are used for training, while class-II polymers are used for testing. The structure variation requires the model to learn generalizable dynamics. (b) R_g^2 for the three training polymers with smallest, median, and largest $\langle R_g^2\rangle$, over a 300k τ period. Our training trajectories are 50k τ long (black dashed line), while we use 5M τ long trajectories for evaluation. (c) Short ground truth MD of training trajectory length gives high-variance, poor estimation of $\langle R_g^2\rangle$. (d) Example polymer before and after coarse-graining. The green beads and bonds represent the CG structure. (e, f, g) $\langle R_g^2\rangle$ estimation performance of the supervised learning baselines using (e) GNN, (f) LSTM, and (g) our learned simulator.

（a）Training和Testing是在同一个化学空间中，通过不同sample得到的
（b）图中的R_g^2 表示聚合物的旋转半径，这与聚合物在溶液中的流变性和致密性有关，对于聚合物的设计很有用
（c）图中的每个点都代表一个system，其中y轴代表的是通过50k τ MD中得到的每个系统的平均旋转半径，用x轴代表50k τ MD得到的每个系统的旋转半径
（e）和（f）的模型是没有预测dynamics的，直接预测R_g^2
（g）作者的方法是预测完系统的dynamics，然后根据这个dynamics trajectory再计算旋转半径

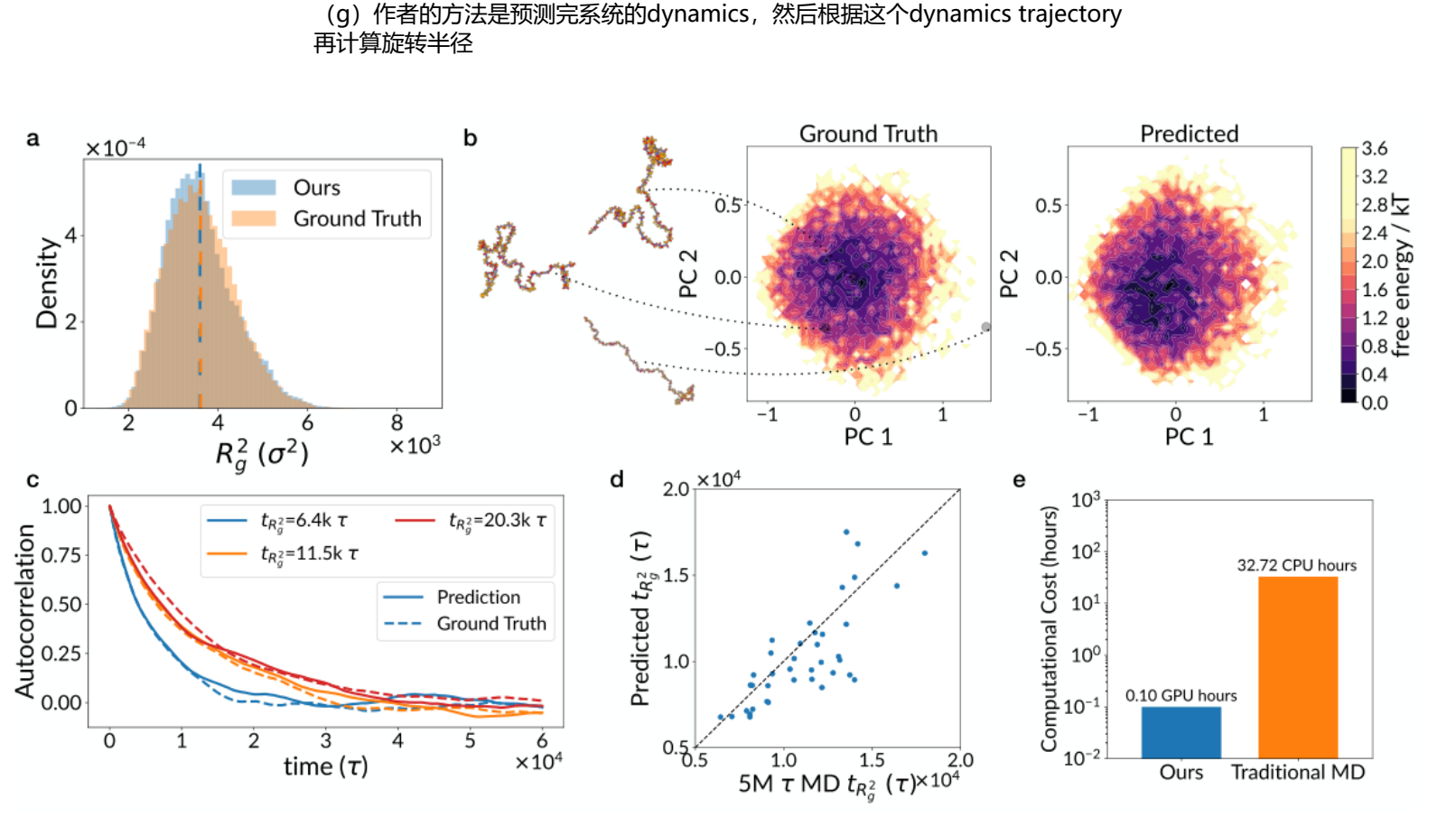


Fig. 3 (a) R_g^2 distribution computed from our learned simulation matches the ground truth. The vertical dashed lines annotate the mean of the distribution. (b) Two-dimensional free energy surface produced with PCA, with representative states with low/high free energy visualized. (c) The autocorrelation function of R_g^2, for the three polymers with the smallest, median, and the largest relaxation time. (d) Prediction performance of our model on the R_g^2 relaxation time. (e) Computational cost comparison of our learned simulation vs. traditional MD. The y-axis is at log-scale.

Relaxation time(弛豫时间): 达到热动平衡所需的时间

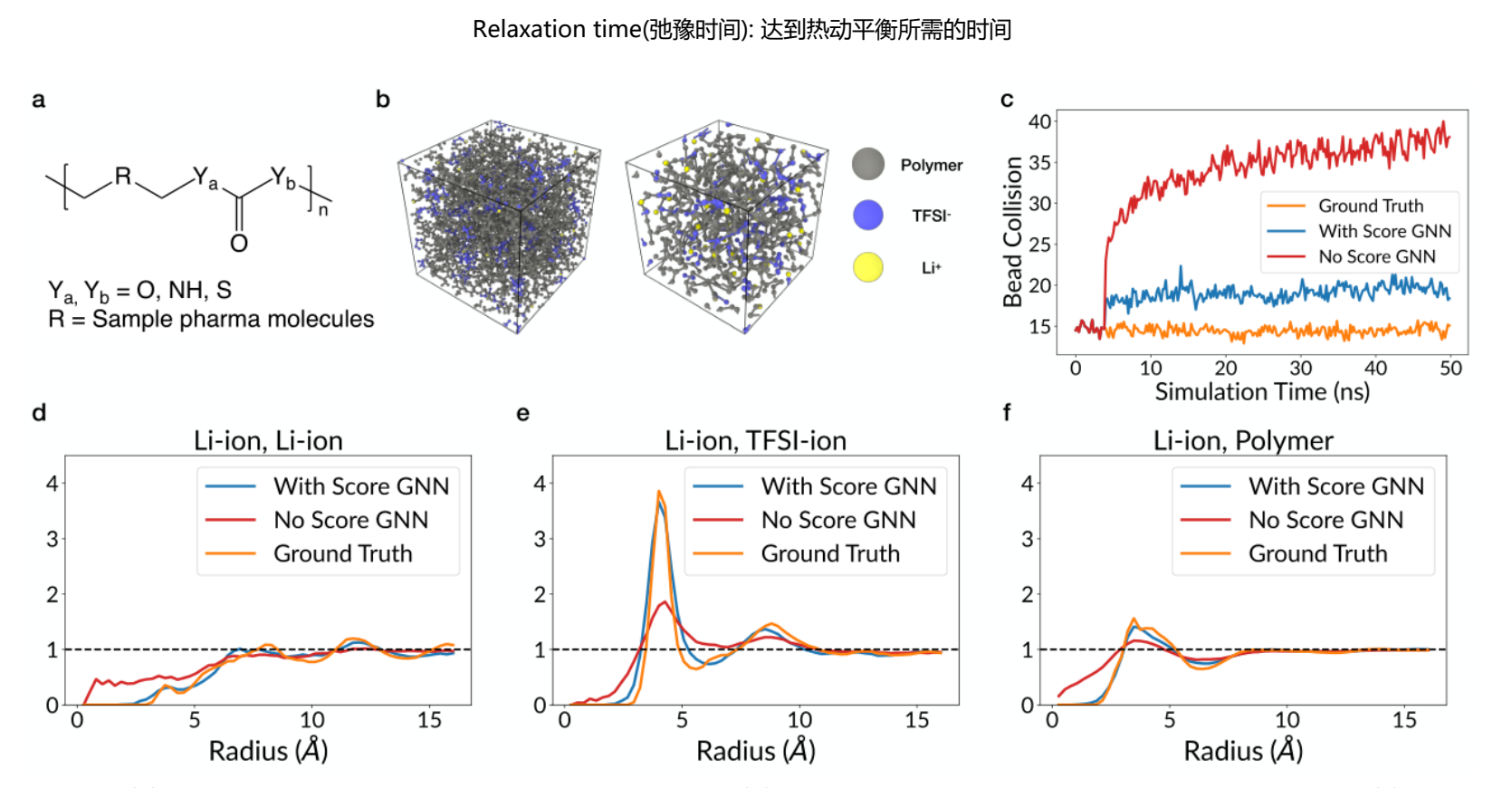


Fig. 4 (a) The chemical space for the polymer in SPEs. (b) Example SPE before and after coarse-graining. (c) Bead collision as a function of simulation time, averaged over the 50 testing SPEs for all methods. (d) RDF of Li-ions, for our model with/without the Score GNN refinement, and the ground truth MD simulation, averaged over a 50-ns trajectory of a selected SPE. (e) RDF of Li-ions and TFSI-ions. (f) RDF of Li-ions and polymer particles.

SPE: Solid polymer electrolyte 固体聚合物电解质
RDF: radial distribution functions

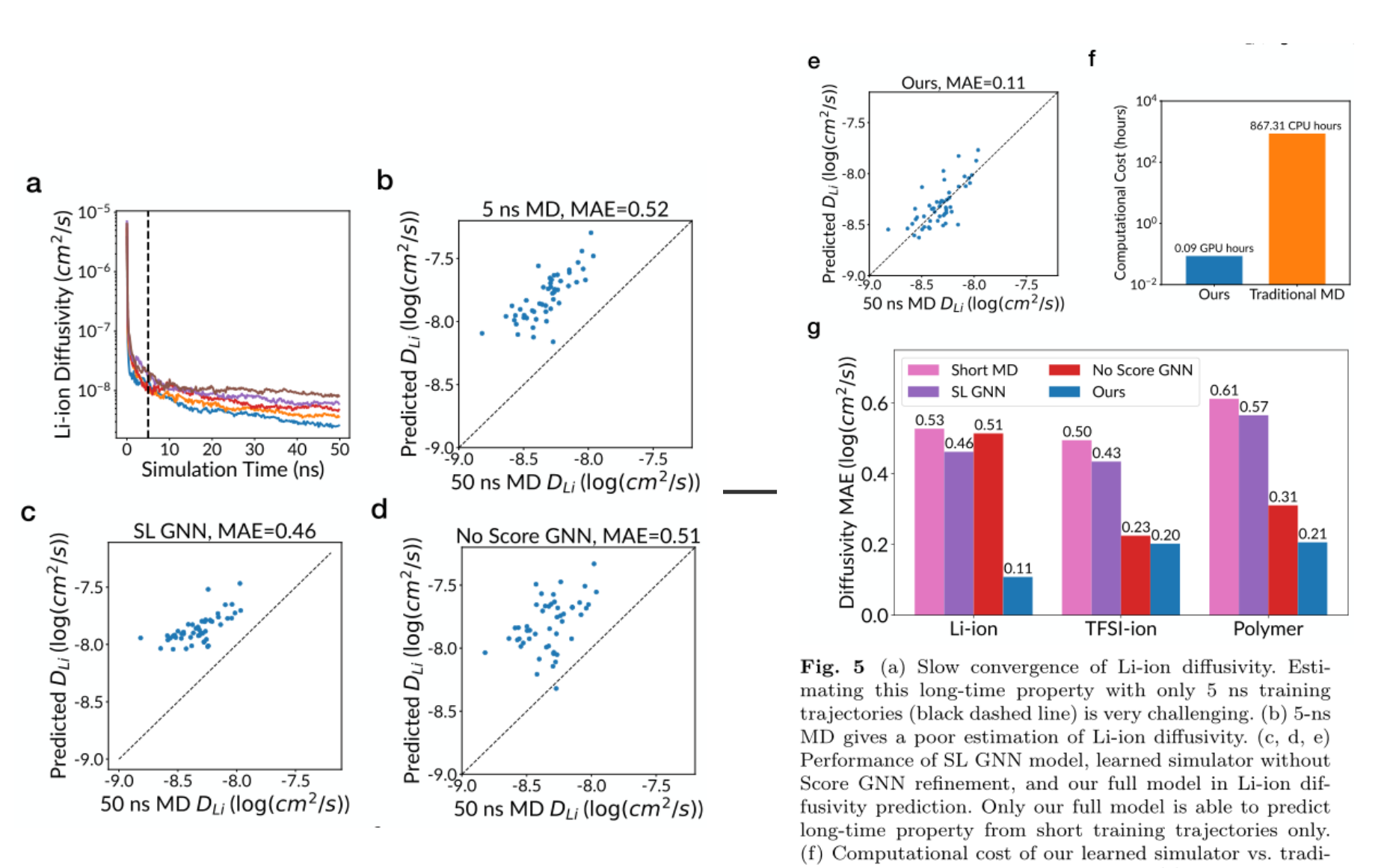


Fig. 5 (a) Slow convergence of Li-ion diffusivity. Estimating this long-time property with only 5 ns training trajectories (black dashed line) is very challenging. (b) 5-ns MD gives a poor estimation of Li-ion diffusivity. (c, d, e) Performance of SL GNN model, learned simulator without Score GNN refinement, and our full model in Li-ion diffusivity prediction. Only our full model is able to predict long-time property from short training trajectories only. (f) Computational cost of our learned simulator vs. traditional MD. The y-axis is at log-scale. (g) Performance of all models in predicting diffusivity of Li-ion, TFSI-ion, and polymer particles.