A Multi-Grained Group Symmetric Framework for Learning Protein-Ligand Binding Dynamics

Shengchao Liu[1,2,*], **Weitao Du**[3,*],
Yanjing Li[4], **Zhuoxinran Li**[5], **Vignesh Bhethanabotla**[2], **Nakul Rampal**[1]
Omar Yaghi[1], **Christian Borgs**[1],
Anima Anandkumar[2], **Hongyu Guo**[6,†], **Jennifer Chayes**[1,†]
[1]UC Berkeley [2]University of CAS [3]Caltech [4]CMU
[5]University of Toronto [6]National Research Council Canada
[*] equal contribution [†] equal advising

Abstract

In drug discovery, molecular dynamics (MD) simulation for protein-ligand binding provides a powerful tool for predicting binding affinities, estimating transport properties, and exploring pocket sites. There has been a long history of improving the efficiency of MD simulations through better numerical methods and, more recently, by augmenting them with machine learning (ML) methods. Yet, challenges remain, such as accurate modeling of extended-timescale simulations. To address this issue, we propose NeuralMD, the first ML surrogate that can facilitate numerical MD and provide accurate simulations of protein-ligand binding dynamics. We propose a principled approach that incorporates a novel physics-informed multi-grained group symmetric framework. Specifically, we propose (1) a BindingNet model that satisfies group symmetry using vector frames and captures the multi-level protein-ligand interactions, and (2) an augmented neural ordinary differential equation solver that learns the trajectory under Newtonian mechanics. For the experiment, we design ten single-trajectory and three multi-trajectory binding simulation tasks. We show the efficiency and effectiveness of NeuralMD, with a 2000$\times$ speedup over standard numerical MD simulation and outperforming all other ML approaches by up to ~80% under the stability metric. We further qualitatively show that NeuralMD reaches more stable binding predictions.

1 Introduction

The simulation of protein-ligand binding dynamics is one of the fundamental tasks in drug discovery [19, 45, 43]. Such simulations of binding dynamics are a key component of the drug discovery pipeline to select, refine, and tailor the chemical structures of potential drugs to enhance their efficacy and specificity. To simulate the protein-ligand binding dynamics, *numerical molecular dynamics (MD)* methods have been extensively developed. However, the numerical MD methods are computationally expensive due to the expensive force calculations on individual atoms in a large protein-ligand system.

To alleviate this issue, *machine learning (ML)* surrogates have been proposed to either augment or replace numerical MD methods to estimate the MD trajectories. However, all prior ML approaches for MD are limited to single-system dynamics (*e.g.*, small molecules or proteins) and not protein-ligand binding dynamics. A primary reason is the lack of large-scale datasets. The first large-scale dataset with binding dynamics was released in May 2023 [35], and to our knowledge, we are now the first to explore it in this paper. Further, prior ML-based MD approaches limit to studying the MD dynamics on a small time interval (1e-15 seconds), while simulation on a longer time interval (*e.g.*, 1e-9 seconds) is needed for specific tasks, such as detecting the transient and

Machine Learning and the Physical Sciences Workshop, NeurIPS 2023.

cryptic states [41] in binding dynamics. However, such longer-time MD simulations are challenging due to the catastrophic buildup of errors over longer rollouts [11].

Another critical aspect that needs to be integrated into ML-based modeling is the group symmetry present in protein-ligand geometry. Specifically, the geometric function should be equivariant to rotation and translation (*i.e.*, SE(3)-equivariance). The principled approach to satisfy equivariance is to use vector frames, which have been previously explored for single molecules [18], but not yet for the protein-ligand binding complexes. The vector frame basis achieves SE(3)-equivariance by projecting vectors (*e.g.*, positions and accelerations) into the vector frame basis, and such a projection can maintain the equivariant property with efficient calculations [23].

Our Approach, NeuralMD. We propose NeuralMD, a multi-grained physics-informed approach designed to handle extended-timestep MD simulations for protein-ligand binding dynamics. Our multi-grained method explicitly decomposes the complexes into three granularities: the atoms in ligands, the backbone structures in proteins, and the residue-atom pairs in the complex, to obtain a scalable approach for modeling a large system. We achieve **group symmetry** in BindingNet through the incorporation of vector frames, and include three levels of vector frame bases for multi-grained modeling, from the atom and backbone level to the residue level for binding interactions.Further, our ML approach NeuralMD preserves the **Newtonian mechanics**. In MD, the movement of atoms is determined by Newton's second law, $F = m \cdot a$, where F is the force, m is the mass, and a is the acceleration of each atom. By integrating acceleration and velocity w.r.t. time, we can obtain the velocities and positions, respectively. Thus in NeuralMD, we formulate the trajectory simulation as a second-order ordinary differential equation (ODE) problem, and it is solved using neural ODE.

Related Work. On one hand, many works have studied the protein-ligand binding problem in the equilibrium state [38, 16, 17, 46], but not the MD simulation for binding dynamics. On the other hand, existing machine learning (ML) methods have studied molecular simulation [47, 6, 27, 12], but they are mainly studying small molecules or proteins. In this work, we propose an ML framework as an MD simulation to learn the protein-ligand binding dynamics.

2 Preliminiaries

Ligands. In this work, we only consider binding complexes with small molecules as ligands. Small molecules can be treated as sets of atoms in the 3D Euclidean space, $\{f^{(l)}, \boldsymbol{x}^{(l)}\}$, where $f^{(l)}$ and $\boldsymbol{x}^{(l)}$ represent the atomic numbers and 3D Euclidean coordinates for atoms in each ligand, respectively.

Proteins. Proteins are macromolecules, which are essentially chains of amino acids or residues. There are 20 natural amino acids, and each amino acid is a small molecule. Noticeably, amino acids are made up of three components: a basic amino group (-NH_2), an acidic carboxyl group (-COOH), and an organic R group (or side chain) that is unique to each amino acid. Additionally, the carbon that connects all three groups is called C_α. (We refer to this Wiki page for more details.) In this work, due to the large volume of atoms in proteins, we will use coarse-grained modelings on proteins and binding complexes. With this regard, the **backbone-level** data structure for each protein is $\{f^{(p)}, \{\boldsymbol{x}_N^{(p)}, \boldsymbol{x}_{C_\alpha}^{(p)}, \boldsymbol{x}_C^{(p)}\}\}$, for the residue type and the coordinates of $N - C_\alpha - C$ in each residue, respectively. (We may ignore the superscript in the coordinates of backbone atoms for brevity since such backbone structures are unique for residues in proteins) In addition to the backbone level, for a coarser-grained data structure, we further consider **residue-level** modeling for binding interactions, $\{f^{(p)}, \boldsymbol{x}^{(p)}\}$, where the coordinate of C_α is treated as the residue-level coordinate, *i.e.*, $\boldsymbol{x}^{(p)} \triangleq \boldsymbol{x}_{C_\alpha}^{(p)}$.

Molecular Dynamics Simulations. Generally, molecular dynamics (MD) describes how each atom in a molecular system moves over time, following Newton's second law of motion:

$$F = m \cdot a = m \cdot \frac{d^2\boldsymbol{x}}{dt^2}, \tag{1}$$

where F is the force, m is the mass, a is the acceleration, $\boldsymbol{x}$ is the position, and t is the time. Then, an MD simulation will take Newtonian dynamics, which is an ordinary differential equation (ODE), to get the trajectories, and such a molecular system can be a small molecule, a protein, a polymer, or a protein-ligand binding complex. The **numerical methods for MD** can be classified into classical MD and *ab-initio* MD, where the difference lies in how the force on each atom is calculated: ab-initio MD calculates the forces using a quantum-mechanics-based method, such as DFT, while classical MD uses an approximated function fit to ab-initio calculations to predict the atomic forces. More

recently, the **machine learning methods for MD** have opened a new perspective by utilizing the group symmetric tools for geometric representation and the neural ODE [4].

Problem Setting: Protein-Ligand Binding Dynamics Simulation. In this work, we focus on simulating the protein-ligand binding dynamics in the semi-flexible setting [31], *i.e.*, proteins with rigid structures and ligands with flexible movements [35, 5]. Thus, the problem is formulated as follows: suppose we have a rigid protein structure $\{f^{(p)}, \{\boldsymbol{x}_N^{(p)}, \boldsymbol{x}_{C_\alpha}^{(p)}, \boldsymbol{x}_C^{(p)}\}\}$ and a ligand with its initial structure and velocity, $\{f^{(l)}, \boldsymbol{x}_0^{(l)}, \boldsymbol{v}_0^{(l)}\}$. We want to predict the trajectories of ligands following the Newtonian dynamics, *i.e.*, the movement of $\{\boldsymbol{x}_t^{(l)}, ...\}$ over time. We also want to clarify two critical points about this problem setting. (1) We consider trajectory prediction, *i.e.*, positions as labels, and no explicit energy and force labels are considered. ML methods for energy prediction followed with numerical ODE solver may require smaller timescales (around 1e-15 seconds), while trajectory prediction, which directly predicts the positions, is agnostic to the magnitude of timescales. This is appealing for datasets like MISATO with larger timescales (1e-9 seconds). (2) Each trajectory is composed of a series of geometries of ligands, and such geometries are called **snapshots**.

3 Method: BindingNet and NeuralMD

In this section, we introduce BindingNet, a multi-grained SE(3)-equivariant geometric model for protein-ligand binding. The input of BindingNet is the geometry of the rigid protein and the ligand at time t, while the output is the force on each atom in the ligand. Our architecture is $SE(3)$ equivariant by extending the equivairant frames for small molecules in [7] to 1. Atom level: $\mathcal{F}_{\text{ligand}}$; 2. Protein level: $\mathcal{F}_{\text{protein}}$; 3. $\mathcal{F}_{\text{complex}}$. See appendix D for the formal definition.

Atom-Level Ligand Modeling. We first generate the atom embedding using one-hot encoding and then aggregate each atom's embedding, $\boldsymbol{z}^{(l)}$, by aggregating all its neighbor's embedding within the cutoff distance c. Then, we obtain the atom's equivariant representation by aggregating its neighborhood's messages as $(\boldsymbol{x}_i^{(l)} - \boldsymbol{x}_j^{(l)}) \cdot \boldsymbol{z}_i^{(l)}$. A subsequent scalarization is carried out based on the atom-level vector frame as $\boldsymbol{h}_{ij}^{(l)} = (\boldsymbol{h}_i^{(l)} \oplus \boldsymbol{h}_j^{(l)}) \cdot \mathcal{F}_{\text{ligand}}$, where $\oplus$ is the concatenation. Finally, it is passed through several equivariant message-passing layers (MPNN) defined as:

$$\text{vec}_i^{(l)} = \text{vec}_i^{(l)} + \text{Agg}_j\big(\text{vec}_i^{(l)} \cdot \text{MLP}(\boldsymbol{h}_{ij}) + (\boldsymbol{x}_i^{(l)} - \boldsymbol{x}_j^{(p)}) \cdot \text{MLP}(\boldsymbol{h}_{ij})\big), \tag{2}$$

where $\text{MLP}(\cdot)$ and $\text{Agg}(\cdot)$ are the multi-layer perceptron and mean aggregation functions, respectively. $\text{vec} \in \mathbb{R}^3$ is a vector assigned to each atom and is initialized as 0. The outputs are atom representation and vector ($\boldsymbol{h}^{(l)}$ and $\text{vec}^{(l)}$), and they are passed to the complex module.

Backbone-Level Protein Modeling. For the coarse-grained modeling of proteins, we consider three backbone atoms in each residue. We first obtain the atom embedding on three atom types, and then we obtain each atom's representation $\boldsymbol{z}^{(p)}$ by aggregating its neighbor's representation. Then, we obtain an equivariant atom representation by aggregating the edge information, $(\boldsymbol{x}_i^{(p)} - \boldsymbol{x}_j^{(p)}) \cdot \boldsymbol{z}_i^{(p)}$, within cutoff distance c. Following which is the scalarization on the residue frame $\boldsymbol{h}_{ij}^{(p)} = (\boldsymbol{h}_i^{(p)} \oplus \boldsymbol{h}_j^{(p)}) \cdot \mathcal{F}_{\text{protein}}$. Recall that we also have the residue type, and with a type embedding $\tilde{\boldsymbol{z}}^{(p)}$, we can obtain the final residue-level representation using an MPNN layer as $\boldsymbol{h}^{(p)} = \tilde{\boldsymbol{z}}^{(p)} + (\boldsymbol{h}_{N,C_\alpha}^{(p)} + \boldsymbol{h}_{C_\alpha,C}^{(p)})/2$. We leave the **Residue-Level Complex Modeling** in appendix F.

NODE for molecular dynamics As clarified in Section 2, molecular dynamics follows Newtonian dynamics, and we solve it as an ordinary differential equation (ODE) problem. The BindingNet takes in the molecular system geometry $(\boldsymbol{x}_t^{(l)}, \boldsymbol{x}^{(p)})$ at arbitrary time t, and outputs the forces.

To learn the MD trajectory following second-order ODE, we propose the following formulation of the second-order ODE within one integration call:

$$\begin{bmatrix} d\boldsymbol{x}/dt \\ d\boldsymbol{v}/dt \end{bmatrix} = \begin{bmatrix} \boldsymbol{v} \\ F/m \end{bmatrix}, \tag{3}$$

where F is the output forces from BindingNet. This means we augment ODE derivative space by concurrently calculating the accelerations and velocities, allowing simultaneous integration of velocities

Table 1: Results on ten single-trajectory binding dynamics prediction. Four evaluation metrics are considered: MAE (Å, ↓), MSE (↓), and Stability (%, ↓).

PDB ID	Metric	VerletMD	GNN-MD	DenoisingLD	NeuralMD (Ours)
5WIJ	MAE	9.618	2.319	2.254	**2.118**
	MSE	6.401	1.553	1.502	**1.410**
	Stability	79.334	45.369	18.054	**12.654**
4ZX0	MAE	21.033	2.255	1.998	**1.862**
	MSE	14.109	1.520	1.347	**1.260**
	Stability	76.878	41.332	23.267	**18.189**
3EOV	MAE	25.403	3.383	3.505	**3.287**
	MSE	17.628	2.332	2.436	**2.297**
	Stability	91.129	57.363	51.590	**44.775**
4K6W	MAE	14.682	3.674	3.555	**3.503**
	MSE	9.887	2.394	2.324	**2.289**
	Stability	87.147	57.852	39.580	**38.562**
1KTI	MAE	18.067	**6.534**	6.657	6.548
	MSE	12.582	4.093	4.159	**4.087**
	Stability	77.315	4.691	7.377	**0.525**

PDB ID	Metric	VerletMD	GNN-MD	DenoisingLD	NeuralMD (Ours)
1XP6	MAE	13.444	2.303	1.915	**1.778**
	MSE	9.559	1.505	1.282	**1.182**
	Stability	86.393	43.019	28.417	**19.256**
4YUR	MAE	15.674	7.030	6.872	**6.807**
	MSE	10.451	4.662	4.520	**4.508**
	Stability	81.309	50.238	32.423	**23.250**
4G3E	MAE	5.181	2.672	2.577	**2.548**
	MSE	3.475	1.743	1.677	**1.655**
	Stability	65.377	16.365	7.188	**2.113**
6B7F	MAE	31.375	4.129	3.952	**3.717**
	MSE	21.920	2.759	2.676	**2.503**
	Stability	87.550	54.900	16.050	**3.625**
3B9S	MAE	19.347	2.701	2.464	**2.351**
	MSE	11.672	1.802	1.588	**1.527**
	Stability	41.667	43.889	8.819	**0.000**

and positions. Ultimately, following Newtonian mechanics, the coordinates at time t are integrated as:

$$\begin{aligned} F_\tau^{(l)} &= \text{BindingNet}(f^{(l)}, \boldsymbol{x}_\tau^{(l)}, f^{(p)}, \boldsymbol{x}_N^{(p)}, \boldsymbol{x}_{C_\alpha}^{(p)}, \boldsymbol{x}_C^{(p)}), \qquad \boldsymbol{a}_\tau^{(l)} = \frac{F_\tau^{(l)}}{m}, \\ \hat{\boldsymbol{v}}_t^{(l)} &= \boldsymbol{v}_0^{(l)} + \int_0^t \boldsymbol{a}_\tau^{(l)} d\tau, \qquad \hat{\boldsymbol{x}}_t^{(l)} = \boldsymbol{x}_0^{(l)} + \int_0^t \hat{\boldsymbol{v}}_\tau^{(l)} d\tau. \end{aligned} \tag{4}$$

The objective is the mean absolute error between the predicted coordinates and ground-truth coordinates: $\mathcal{L} = \mathbb{E}_t\big[|\hat{\boldsymbol{x}}_t^{(l)} - \boldsymbol{x}_t^{(l)}|\big]$. An illustration of NeuralMD pipeline is in appendix Figure 2.

4 Experiments

Datasets. We consider MISATO in our work [35]. It is built on 16,972 experimental protein-ligand complexes extracted from the protein data bank (PDB) [2]. For each protein-ligand complex, the trajectory comprises 100 snapshots in 8 nanoseconds under the fixed temperature and pressure. We want to highlight that MD trajectories allow the analysis of small-range structural fluctuations of the protein-ligand complex. See appendix E for the basic statistics.

Experiments Settings. We consider two experiment settings. The first type of experiment is the single-trajectory prediction, where both the training and test data are snapshots from the same trajectory, and they are divided temporally. The second type of experiment is the multi-trajectory prediction, where each data point is the sequence of all the snapshots from one trajectory, and the training and test data correspond to different sets of trajectories. The first type of task has been widely studied in the existing literature for other molecular systems. Specifically, both the training and test data are from the same trajectory of one single protein-ligand complex, and here we take the first 80 snapshots for training and the remaining 20 snapshots for test.

Evaluation Metrics. For MD simulation, the evaluation is a critical factor for evaluating trajectory prediction. For both experiment settings, the trajectory recovery is the most straightforward evaluation metric. To evaluate this, we take both the mean absolute error (MAE) and mean squared error (MSE) between the predicted coordinates and ground-truth coordinates. Stability is also an important metric for evaluating the predicted MD trajectory. The intuition is that the prediction on long-time MD trajectory can enter a pathological state (*e.g.*, bond breaking), and stability is the measure to quantify such observation. It is defined as $\mathbb{P}_{i,j}\big|\|\boldsymbol{x}_i - \boldsymbol{x}_j\| - \boldsymbol{b}_{i,j}\big| > \Delta$, where $\boldsymbol{b}_{i,j}$ is the pair distance at the last snapshot (the most equilibrium state), and we take $\Delta = 0.5$ Å. Another metric considered is frames per second (FPS) on a single Nvidia-V100 GPU card, and it measures the MD efficiency.

Results on single-trajectory prediction Results are in Table 1 (baselines are introduced in appendix G). The first observation is that the baseline VertletMD has a clear performance gap compared to the other methods. This verifies that using ML models to predict the energy (or force) at each snapshot, and then using a numerical integration algorithm can fail in the long-time simulations [11]. Additionally, we can observe that the recovery error of trajectory (MAE and MSE) occasionally fails to offer a discernible distinction among methods (*e.g.*, for protein-ligand complex 3EOV, 1KT1, and 4G3E), though NeuralMD is slightly better. However, the stability (%) can be a distinctive factor in method comparisons, where we observe NeuralMD outperform on all 10 tasks up to ~80%. More detailed analysis is provided in appendix G.

References

[1] Marloes Arts, Victor Garcia Satorras, Chin-Wei Huang, Daniel Zugner, Marco Federici, Cecilia Clementi, Frank Noe, Robert Pinsler, and Rianne van den Berg. Two for one: Diffusion models and force fields for coarse-grained molecular dynamics. *Journal of Chemical Theory and Computation*, 2023.

[2] Helen M Berman, John Westbrook, Zukang Feng, Gary Gilliland, Talapady N Bhat, Helge Weissig, Ilya N Shindyalov, and Philip E Bourne. The protein data bank. *Nucleic acids research*, 28(1):235–242, 2000.

[3] J Ceriotti. More, and de manolopoulos. *Comput. Phys. Commun*, 185:1019, 2014.

[4] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[5] Gabriele Corso, Hannes Stärk, Bowen Jing, Regina Barzilay, and Tommi Jaakkola. Diffdock: Diffusion steps, twists, and turns for molecular docking. *International Conference on Learning Representations (ICLR)*, 2023.

[6] Stefan Doerr, Maciej Majewsk, Adrià Pérez, Andreas Krämer, Cecilia Clementi, Frank Noe, Toni Giorgino, and Gianni De Fabritiis. Torchmd: A deep learning framework for molecular simulations, 2020.

[7] Weitao Du, He Zhang, Yuanqi Du, Qi Meng, Wei Chen, Nanning Zheng, Bin Shao, and Tie-Yan Liu. Se (3) equivariant graph neural networks with complete local frames. In *International Conference on Machine Learning*, pages 5583–5608. PMLR, 2022.

[8] Hehe Fan, Zhangyang Wang, Yi Yang, and Mohan Kankanhalli. Continuous-discrete convolution for geometry-sequence modeling in proteins. In *The Eleventh International Conference on Learning Representations*, 2022.

[9] Hans Frauenfelder, Guo Chen, Joel Berendzen, Paul W Fenimore, Helén Jansson, Benjamin H McMahon, Izabela R Stroe, Jan Swenson, and Robert D Young. A unified model of protein dynamics. *Proceedings of the National Academy of Sciences*, 106(13):5129–5134, 2009.

[10] Daan Frenkel and Berend Smit. *Understanding molecular simulation: from algorithms to applications*. Academic Press San Diego, 2002.

[11] Xiang Fu, Zhenghao Wu, Wujie Wang, Tian Xie, Sinan Keten, Rafael Gomez-Bombarelli, and Tommi Jaakkola. Forces are not enough: Benchmark and critical evaluation for machine learning force fields with molecular simulations. *arXiv preprint arXiv:2210.07237*, 2022.

[12] Xiang Fu, Tian Xie, Nathan J Rebello, Bradley D Olsen, and Tommi Jaakkola. Simulate time-integrated coarse-grained molecular dynamics with geometric machine learning. *arXiv preprint arXiv:2204.10348*, 2022.

[13] Joe G Greener and David T Jones. Differentiable molecular simulation can learn all the parameters in a coarse-grained force field for proteins. *PloS one*, 16(9):e0256990, 2021.

[14] Jiaqi Guan, Wesley Wei Qian, Xingang Peng, Yufeng Su, Jian Peng, and Jianzhu Ma. 3d equivariant diffusion for target-aware molecule generation and affinity prediction. *arXiv preprint arXiv:2303.03543*, 2023.

[15] John Ingraham, Vikas Garg, Regina Barzilay, and Tommi Jaakkola. Generative models for graph-based protein design. *Advances in neural information processing systems*, 32, 2019.

[16] José Jiménez, Miha Skalic, Gerard Martinez-Rosell, and Gianni De Fabritiis. K deep: protein–ligand absolute binding affinity prediction via 3d-convolutional neural networks. *Journal of chemical information and modeling*, 58(2):287–296, 2018.

[17] Derek Jones, Hyojin Kim, Xiaohua Zhang, Adam Zemla, Garrett Stevenson, WF Drew Bennett, Daniel Kirshner, Sergio E Wong, Felice C Lightstone, and Jonathan E Allen. Improved protein–ligand binding affinity prediction with structure-based deep fusion inference. *Journal of chemical information and modeling*, 61(4):1583–1592, 2021.

[18] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[19] Visvaldas Kairys, Lina Baranauskiene, Migle Kazlauskiene, Daumantas Matulis, and Egidijus Kazlauskas. Binding affinity in drug design: experimental and computational techniques. *Expert opinion on drug discovery*, 14(8):755–768, 2019.

[20] Martin Karplus and John Kuriyan. Molecular dynamics and protein function. *Proceedings of the National Academy of Sciences*, 102(19):6679–6685, 2005.

[21] Johannes Klicpera, Shankari Giri, Johannes T Margraf, and Stephan Günnemann. Fast and uncertainty-aware directional message passing for non-equilibrium molecules. *arXiv preprint arXiv:2011.14115*, 2020.

[22] Divya B Korlepara, CS Vasavi, Shruti Jeurkar, Pradeep Kumar Pal, Subhajit Roy, Sarvesh Mehta, Shubham Sharma, Vishal Kumar, Charuvaka Muvva, Bhuvanesh Sridharan, et al. Plas-5k: Dataset of protein-ligand affinities from molecular dynamics for machine learning applications. *Scientific Data*, 9(1):548, 2022.

[23] Shengchao Liu, Weitao Du, Yanjing Li, Zhuoxinran Li, Zhiling Zheng, Chenru Duan, Zhiming Ma, Omar Yaghi, Anima Anandkumar, Christian Borgs, Jennifer Chayes, Hongyu Guo, and Jian Tang. Symmetry-informed geometric representation for molecules, proteins, and crystalline materials. *arXiv preprint arXiv:2306.09375*, 2023.

[24] Shengchao Liu, Weitao Du, Zhi-Ming Ma, Hongyu Guo, and Jian Tang. A group symmetric stochastic differential equation model for molecule multi-modal pretraining. In *International Conference on Machine Learning*, pages 21497–21526. PMLR, 2023.

[25] Albert Musaelian, Simon Batzner, Anders Johansson, Lixin Sun, Cameron J Owen, Mordechai Kornbluth, and Boris Kozinsky. Learning local equivariant representations for large-scale atomistic dynamics. *arXiv preprint arXiv:2204.05249*, 2022.

[26] Albert Musaelian, Simon Batzner, Anders Johansson, Lixin Sun, Cameron J Owen, Mordechai Kornbluth, and Boris Kozinsky. Learning local equivariant representations for large-scale atomistic dynamics. *Nature Communications*, 14(1):579, 2023.

[27] Albert Musaelian, Anders Johansson, Simon Batzner, and Boris Kozinsky. Scaling the leading accuracy of deep equivariant models to biomolecular simulations of realistic size. *arXiv preprint arXiv:2304.10061*, 2023.

[28] Xingang Peng, Shitong Luo, Jiaqi Guan, Qi Xie, Jian Peng, and Jianzhu Ma. Pocket2mol: Efficient molecular sampling based on 3d protein pockets. In *International Conference on Machine Learning*, pages 17644–17655. PMLR, 2022.

[29] U Deva Priyakumar, Divya B Korlepara, CS Vasavi, Rakesh Srivastava, Pradeep Kumar Pal, Saalim H Raza, Vishal Kumar, Shivam Pandit, Aathira G Nair, Sanjana Pandey, et al. Plas-20k: Extended dataset of protein-ligand affinities from md simulations for machine learning applications. 2023.

[30] Dennis C Rapaport. *The art of molecular dynamics simulation*. Cambridge university press, 2004.

[31] Veronica Salmaso and Stefano Moro. Bridging molecular docking to molecular dynamics in exploring ligand-protein recognition process: An overview. *Frontiers in pharmacology*, 9:923, 2018.

[32] Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) equivariant graph neural networks. *arXiv preprint arXiv:2102.09844*, 2021.

[33] Kristof T Schütt, Huziel E Sauceda, P-J Kindermans, Alexandre Tkatchenko, and K-R Müller. Schnet–a deep learning architecture for molecules and materials. *The Journal of Chemical Physics*, 148(24):241722, 2018.

[34] Kristof T Schütt, Oliver T Unke, and Michael Gastegger. Equivariant message passing for the prediction of tensorial properties and molecular spectra. *arXiv preprint arXiv:2102.03150*, 2021.

[35] Till Siebenmorgen, Filipe Menezes, Sabrina Benassou, Erinc Merdivan, Stefan Kesselheim, Marie Piraud, Fabian J Theis, Michael Sattler, and Grzegorz M Popowicz. Misato-machine learning dataset of protein-ligand complexes for structure-based drug discovery. *bioRxiv*, pages 2023–05, 2023.

[36] Tess Smidt, Nathaniel Thomas, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

[37] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[38] Marta M Stepniewska-Dziubinska, Piotr Zielenkiewicz, and Pawel Siedlecki. Development and evaluation of a deep learning model for protein–ligand binding affinity prediction. *Bioinformatics*, 34(21):3666–3674, 2018.

[39] Philipp Tholke and Gianni De Fabritiis. Equivariant transformers for neural network based molecular potentials. In *International Conference on Learning Representations*, 2022.

[40] Aidan P Thompson, H Metin Aktulga, Richard Berger, Dan S Bolintineanu, W Michael Brown, Paul S Crozier, Pieter J in't Veld, Axel Kohlmeyer, Stan G Moore, Trung Dac Nguyen, et al. Lammps-a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. *Computer Physics Communications*, 271:108171, 2022.

[41] Sandor Vajda, Dmitri Beglov, Amanda E Wakefield, Megan Egbert, and Adrian Whitty. Cryptic binding sites on proteins: definition, detection, and druggability. *Current opinion in chemical biology*, 44:1–8, 2018.

[42] David Van Der Spoel, Erik Lindahl, Berk Hess, Gerrit Groenhof, Alan E Mark, and Herman JC Berendsen. Gromacs: fast, flexible, and free. *Journal of computational chemistry*, 26(16):1701–1718, 2005.

[43] Mikhail Volkov, Joseph-André Turk, Nicolas Drizard, Nicolas Martin, Brice Hoffmann, Yann Gaston-Mathé, and Didier Rognan. On the frustration to predict binding affinities from protein–ligand structures with deep neural networks. *Journal of medicinal chemistry*, 65(11):7946–7958, 2022.

[44] Fang Wu and Stan Z Li. Diffmd: A geometric diffusion model for molecular dynamics simulations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 5321–5329, 2023.

[45] Jincai Yang, Cheng Shen, and Niu Huang. Predicting or pretending: artificial intelligence for protein-ligand interactions lack of sufficiently large and unbiased datasets. *Frontiers in pharmacology*, 11:69, 2020.

[46] Ziduo Yang, Weihe Zhong, Qiujie Lv, Tiejun Dong, and Calvin Yu-Chian Chen. Geometric interaction graph neural network for predicting protein–ligand binding affinities from 3d structures (gign). *The Journal of Physical Chemistry Letters*, 14(8):2020–2033, 2023.

[47] Linfeng Zhang, Jiequn Han, Han Wang, Roberto Car, and EJPRL Weinan. Deep potential molecular dynamics: a scalable model with the accuracy of quantum mechanics. *Physical review letters*, 120(14):143001, 2018.

[48] Xuan Zhang, Limei Wang, Jacob Helwig, Youzhi Luo, Cong Fu, Yaochen Xie, Meng Liu, Yuchao Lin, Zhao Xu, Keqiang Yan, et al. Artificial intelligence for science in quantum, atomistic, and continuum systems. *arXiv preprint arXiv:2307.08423*, 2023.

A Visual Analysis

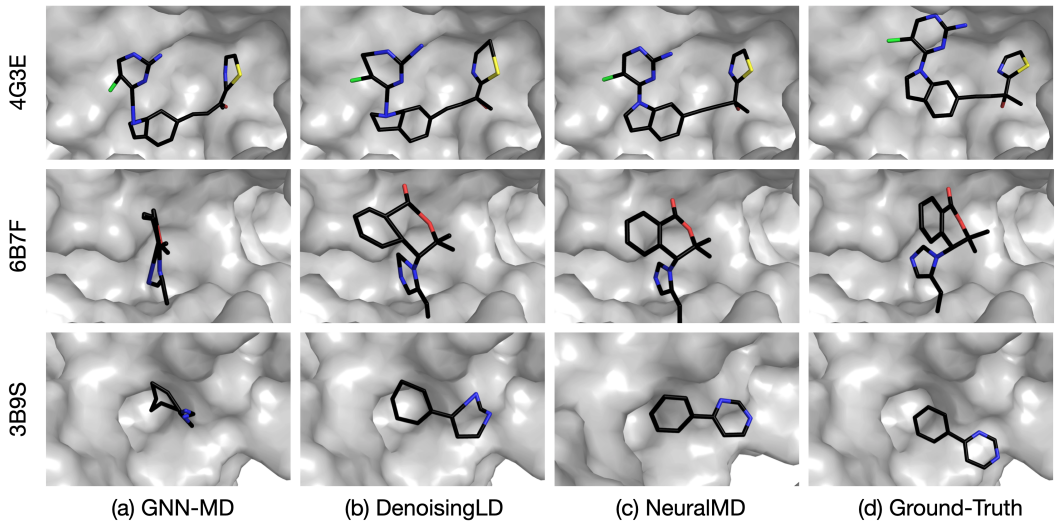


Figure 1: Visualization of last-snapshot binding predictions on three PDB complexes. NeuralMD stays more stable than DenoisingLD, exhibiting a lower degree of torsion with the natural conformations. Other methods collapse heavily, including GNN-MD and VerletMD, where atoms extend beyond the frame for the latter.

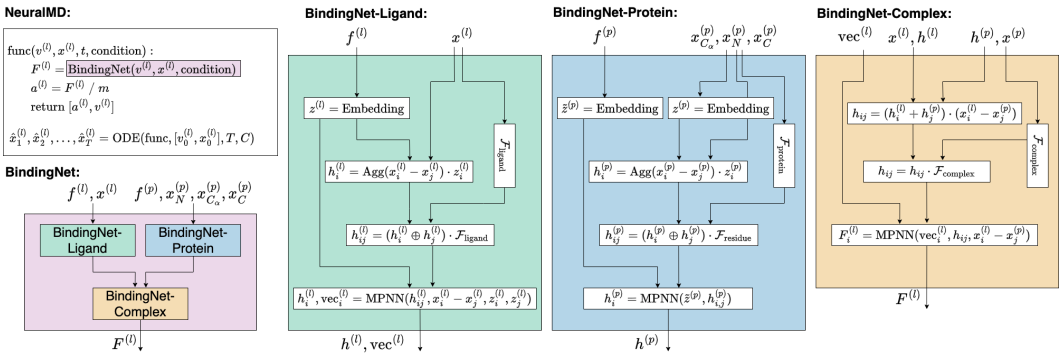


Figure 2: Brief pipeline of NeuralMD. In the three key modules of BindingNet, there are three vertical boxes, corresponding to three granularities of vector frames, as in Equations (22) to (24). More details are in Appendix F.

B Related Works and Preliminaries

SE(3)-Equivariant Representation for Small Molecules and Proteins. The molecular systems are indeed a set of atoms located in the 3D Euclidean space. From a machine learning point of view, the representation function or encoding function of such molecular systems needs to be group-symmetric, *i.e.*, the representation needs to be equivariant when we rotate or translate the whole system. Such symmetry is called the SE(3)-equivariance. Recently works [23, 48] on molecules has provided a unified way of equivariant geometric modeling. They categorize the mainstream representation methods into three big venues: SE(3)-invariant models, SE(3)-equivariant models with spherical frame basis, and SE(3)-equivariant models with vector frame basis. (1) Invariant models that utilize invariant features (distances and angles) to predict the energies [33, 21], but the derived forces are challenging for ML optimization after integration. (2) Equivariant models with spherical frames that include a computationally expensive tensor-product operation [36, 25], which is unsuitable for large molecular systems. (3) Equivariant models with vector frames that have been explored for single stable molecules, including molecule representation and pretraining [32, 34, 24], molecule conformation generation [7], protein representation [8], and protein folding and design [15, 18]. However, no one has tried it for binding complexes.

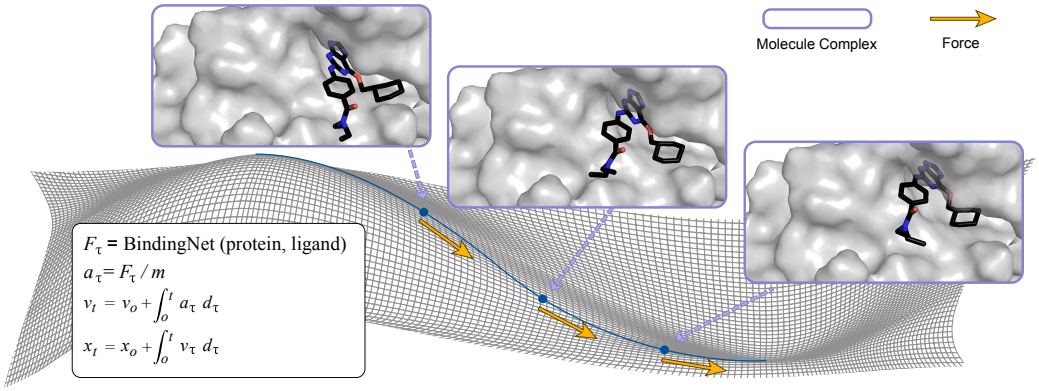


Figure 3: Illustrations of NeuralMD for protein-ligand binding dynamics. The landscape depicts the energy level, and the binding dynamic leads to an equilibrium state with lower energy.For NeuralMD, the force prediction follows the SE(3)-equivariance, and the dynamics prediction follows the Newtonian mechanics.

Structure-Based Ligand Pose Generation in Equilibrium State. One seemingly related but fundamentally different research line is the structure-based ligand pose generation. The core idea is to generate the ligand pose based on the protein structure. Statistical generative methods like autoregressive [28] and denoising diffusion [5, 14] have been applied to this task. We want to highlight that these methods generate a trajectory, but they are essentially different for two reasons: (1) These works consider only the conformations at the equilibrium state, and no dynamics information is considered. (2) The inference of denoising diffusion generation is in the form of Monte Carlo (MC), which formulates a trajectory for each sampled sample. However, such trajectories include no physical information (*e.g.*, energy, and force), while the movements of molecules follow the Newtonian mechanics (*e.g.*, force fields) in MD simulation.

Potential Energy and Force Learning for Molecular Dynamics Simulation. One straightforward way of molecular dynamics (MD) simulation is through potential energy modeling. *Numerical methods* for MD simulation can be classified into classical MD and *ab-initio* MD, depending on using the classical mechanism or quantum mechanism to calculate the forces. Alternatively, a machine learning (ML) research line is to adopt geometric representation methods to learn the energies or the forces, *e.g.*, by the geometric methods listed above. The first work is DeePMD [47], which targets learning the potential energy function at each conformation. For inference, the learned energy can be applied to update the atom placement using i-PI software [3], composing the MD trajectories. TorchMD [6] utilizes TorchMD-Net [39] for energy prediction, which will be fed into the velocity Verlet algorithm for MD simulation. Similarly, **(author?)** adopts Allegro [26] model to learn the force at each conformation. The learned model will be used for MD trajectory simulation using LAMMPS [40]. In theory, all the geometric models on small molecules [32, 34, 24] and proteins [8] can be applied to the MD simulation task. However, there are two main challenges: (1) They require the interval between snapshots to be at the femtoseconds (1e-15 seconds) level for integration to effectively capture the motion of the molecules. (2) They take the position-energy pairs independently, and thus, they ignore their temporal correlations during learning.

Trajectory Learning for Molecular Dynamics Estimation. More recent works have explored MD simulation by directly learning the placement along the trajectories. There are two key differences between energy and trajectory prediction for MD: (1) Energy prediction takes each conformation and energy as IID, while trajectory learning optimizes the conformations along the whole trajectory, enforcing the temporal relation. (2) The time interval of trajectory learning is agnostic to the time interval, and energy prediction can be sensitive to longer MD simulations. More concretely, along such trajectory learning research line, CGDMS [13] builds an SE(3)-invariant model, followed by the velocity Verlet algorithm for MD simulation. DiffMD [44] is a Markovian method and treats the dynamics between two consecutive snapshots as a coordinate denoising process. It then applies the SDE solver [37] to solve the molecular dynamics. A parallel work, DFF [1], applies a similar idea for MD simulation. CG-MD [12] encodes a hierarchical graph neural network model for an auto-regressive position generation and then adopts the denoising method for fine-tuning. However, these works disregard the prior knowledge of the Newtonian mechanics governing the motion of atoms.

Protein-Ligand Binding Dynamics. The MD simulation papers discussed so far are mainly for small molecules or proteins, not the binding dynamics. On the other hand, many works have studied the protein-ligand binding problem in the equilibrium state [38, 16, 17, 46], but not the dynamics. In this work, we consider a more challenging task, which is the protein-ligand binding dynamics. The viability of this work is also attributed to the efforts of the scientific community, where more binding dynamics dataset has been revealed, including PLAS-5k [22], MISATO [35], and PLAS-20k [29].

Preliminary on Molecular Dynamics. Molecular dynamics (MD) simulations predict how every atom in a molecular system moves over time, which is determined by the interatomic interactions following Newton's

Table 2: Comparison of different numerical and machine learning (ML) methods for molecular dynamics (MD). AR for autoregressive and denoising for denoising diffusion method.

Category	Method	Energy / Force Calculation	Dynamics	Objective Function	Publications
Numerical Methods	Classical MD	Classical Mechanics: Force Field	Newtonian Dynamics	–	–
	Ab-initio MD	Quantum Mechanics: DFT for Schrodinger Equation	Newtonian Dynamics	–	–
	Langevin MD	Classical Mechanics: Force Field	Langevin Dynamics	–	–
ML Methods	DeePMD [47]	Atom-level Modeling	Newtonian Dynamics (i-PI)	Energy Prediction	PRL'18
	TorchMD [6]	Atom-level Modeling	Newtonian Dynamics (velocity Verlet)	Energy Prediction	ACS'20
	Allegro-LAMMPS [27]	Atom-level Modeling	Newtonian Dynamics (LAMMPS)	Force Prediction	ArXiv'23
	VerletMD (Ours, baseline)	Atom-level Modeling	Newtonian Dynamics (velocity Verlet)	Energy Prediction	–
	CGDMS [13]	Atom-level Modeling	Newtonian Dynamics (velocity Verlet)	Position Prediction	PLOS'21
	DiffMD [44]	Atom-level Modeling	AR + Denoising	Position Prediction	AAAI'23
	DFF [1]	Atom-level Modeling	AR + Denoising	Position Prediction	ACS'23
	CG-MD [12]	Atom-level Modeling	AR + Denoising	Position Prediction	TMLR'23
	LigandMD (Ours, baseline)	Atom-level Modeling	AR + Denoising	Position Prediction	–
	NeuralMD (Ours)	Atom-level Modeling	Newtonian Dynamics (NODE)	Position Prediction	–

second law. Such a molecular system includes Small Molecules [30, 47], proteins [20, 9, 1], polymers [12], and protein-ligand complexes [22, 35]. Typically, an MD simulation is composed of two main steps, *i.e.*, (1) the energy and force calculation and (2) integration of the equations of motion governed by Newton's second law of motion, using the initial conditions and forces calculated in step (1). As the initial condition, the initial positions and velocities are given for all the particles (*e.g.*, atoms) in the molecular system; the MD simulation repeats the two steps to get a trajectory. Such MD simulations can be used to calculate the equilibrium and transport properties of molecules, materials, and biomolecular systems [10].

In such an MD simulation, one key factor is estimating the forces on each atom. The function that gives the energy of a molecular system as a function of its structure (and forces via the gradient of the energy with respect to those atomic coordinates) is referred to as a potential energy surface (PES). In general, MD simulations integrate the equations of motion using a PES from one of two sources: (1) Classical MD using the force fields, which are parameterized equations that approximate the true PES, and are less costly to evaluate, allowing for the treatment of larger systems and longer timesteps. (2) *ab-inito* MD (which calculates the energy of a molecular system via electronic structure methods, *e.g.*, DFT) provide more accurate PES, but are limited in the system size and timesteps that are practically accessible due to the cost of evaluating the PES at a given point.

C Group Symmetry and Equivariance

In this article, a 3D molecular graph is represented by a collection of 3D point clouds. The corresponding symmetry group is SE(3), which consists of translations and rotations. Recall that we define the notion of equivariance functions in $\mathbf{R}^3$ in the main text through group actions. Formally, the group SE(3) is said to act on $\mathbf{R}^3$ if there is a mapping $\phi : \text{SE(3)} \times \mathbf{R}^3 \to \mathbf{R}^3$ satisfying the following two conditions:

1. if $e \in \text{SE(3)}$ is the identity element, then
$$\phi(e, \boldsymbol{r}) = \boldsymbol{r} \quad \text{for } \forall \boldsymbol{r} \in \mathbf{R}^3.$$
2. if $g_1, g_2 \in \text{SE(3)}$, then
$$\phi(g_1, \phi(g_2, \boldsymbol{r})) = \phi(g_1 g_2, \boldsymbol{r}) \quad \text{for } \forall \boldsymbol{r} \in \mathbf{R}^3.$$

Then, there is a natural SE(3) action on vectors $\boldsymbol{r}$ in $\mathbf{R}^3$ by translating $\boldsymbol{r}$ and rotating $\boldsymbol{r}$ for multiple times. For $g \in \text{SE(3)}$ and $\boldsymbol{r} \in \mathbf{R}^3$, we denote this action by $g\boldsymbol{r}$. Once the notion of group action is defined, we say a function $f : \mathbf{R}^3 \to \mathbf{R}^3$ that transforms $\boldsymbol{r} \in \mathbf{R}^3$ is equivariant if:

$$f(g\boldsymbol{r}) = gf(\boldsymbol{r}), \quad \text{for } \forall \ \boldsymbol{r} \in \mathbf{R}^3.$$

On the other hand, $f : \mathbf{R}^3 \to \mathbf{R}^1$ is invariant, if f is independent of the group actions:

$$f(g\boldsymbol{r}) = f(\boldsymbol{r}), \quad \text{for } \forall \ \boldsymbol{r} \in \mathbf{R}^3.$$

For some scenarios, our problem is chiral sensitive. That is, after mirror reflecting a 3D molecule, the properties of the molecule may change dramatically. In these cases, it's crucial to include reflection transformations into consideration. More precisely, we say an SE(3) equivariant function f is **reflection anti-symmetric**, if:

$$f(\rho\boldsymbol{r}) \neq f(\boldsymbol{r}), \tag{5}$$

for reflection $\rho \in \text{E(3)}$.

D Equivariant Vector Frames

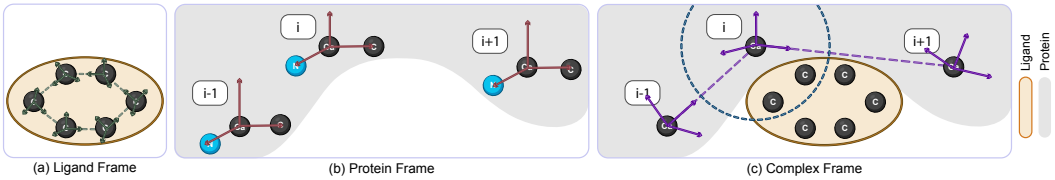


Figure 4: Three granularities of vector frame basis in BindingNet: (a) atom-level basis for ligands, (b) backbone-level basis for proteins, and (c) residue-level basis for the protein-ligand complex.

Frame is a popular terminology in science areas. In physics, the frame is equivalent to a coordinate system. For example, we may assign a frame to all observers, although different observers may collect different data under different frames, the underlying physics law should be the same. In other words, denote the physics law by f, then f should be an equivariant function.

There are certain ways to choose the frame basis, and below we introduce two main types: the orthogonal basis and the protein backbone basis. The orthogonal basis can be built for flexible 3D point clouds such as atoms, while the protein backbone basis is specifically proposed to capture the protein backbone.

D.1 Basis

Since there are three orthogonal directions in $\mathbf{R}^3$, one natural frame in $\mathbf{R}^3$ can be a frame consisting of three orthogonal vectors:

$$F = (\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3).$$

Once equipped with a frame (coordinate system), we can project all geometric quantities to this frame. For example, an abstract vector $\boldsymbol{x} \in \mathbf{R}^3$ can be written as $\boldsymbol{x} = (r_1, r_2, r_3)$ under the frame F, if: $\boldsymbol{x} = r_1\boldsymbol{e}_1 + r_2\boldsymbol{e}_2 + r_3\boldsymbol{e}_3$. An equivariant frame further requires the three orthonormal vectors in $(\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3)$ to be equivariant. Intuitively, an equivariant frame will transform according to the global rotation or translation of the whole system. Once equipped with an equivariant frame, we can project equivariant vectors into this frame:

$$\boldsymbol{x} = \tilde{r}_1\boldsymbol{e}_1 + \tilde{r}_2\boldsymbol{e}_2 + \tilde{r}_3\boldsymbol{e}_3. \tag{6}$$

We call the process of $\boldsymbol{x} \to \tilde{r} := (\tilde{r}_1, \tilde{r}_2, \tilde{r}_3)$ the **scalarization** or **projection** operation. Since $\tilde{r}_i = \boldsymbol{e}_i \cdot \boldsymbol{x}$ is expressed as an inner product between equivariant vectors, we know that $\tilde{r}$ consists of scalars.

In this article, we assign an equivariant frame to each node/edge, therefore we call them the local frames. Given two atoms with 3D positions $(\boldsymbol{x}_i, \boldsymbol{x}_j)$, we can find the atom (denoted by $\boldsymbol{x}_k$) that is nearest to the center of $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ by KNN algorithms. Then the equivariant frame is defined by:

$$\text{Vector-Frame}(\boldsymbol{x}_i, \boldsymbol{x}_j) := \textbf{Gram-Schmidt}\{\boldsymbol{x}_i - \boldsymbol{x}_j, \boldsymbol{x}_i - \boldsymbol{x}_k, (\boldsymbol{x}_i - \boldsymbol{x}_j) \times (\boldsymbol{x}_i - \boldsymbol{x}_k)\}. \tag{7}$$

The Gram-Schmidt orthogonalization makes sure that the Vector-Frame$(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is orthonormal.

Reflection Antisymmetric Since we implement the cross product $\times$ for building the local frames, the third vector in the frame is a pseudo-vector. Then, the **projection** operation is not invariant under reflections (the inner product between a vector and a pseudo-vector change signs under reflection). Therefore, our model can discriminate two 3D geometries with different chiralities.

Our local frames also enable us to output equivariant vectors by multiplying scalars (v_1, v_2, v_3) with the frame: $\boldsymbol{v} = v_1 \cdot \boldsymbol{e}_1 + v_2 \cdot \boldsymbol{e}_2 + v_3 \cdot \boldsymbol{e}_3$.

Equivariance w.r.t. cross-product The goal is to prove that the cross-product is equivariant to the SE(3)-group, *i.e.*:

$$gx \times gy = g(x \times y), \qquad g \in \text{SE(3)-Group} \tag{8}$$

Geometric proof. From intuition, with rotation matrix g, we are transforming the whole basis, thus the direction of $gx \times gy$ changes equivalently with g. And for the value/length of $gx \times gy$, because $|gx \times gy| = \|gx\| \cdot \|gy\| \cdot \sin\theta = \|x\| \cdot \|y\| \cdot \sin\theta = |x \times y|$. So the length stays the same, and the direction changes equivalently. Intuitively, this interpretation is quite straightforward.

Analytical proof. A more rigorous proof can be found below:

Proof. First, we have that for the rotation matrix g:

$$gx \times gy = \begin{bmatrix} \boldsymbol{g}_1^T \boldsymbol{x} \\ \boldsymbol{g}_2^T \boldsymbol{x} \\ \boldsymbol{g}_3^T \boldsymbol{x} \end{bmatrix} \times \begin{bmatrix} \boldsymbol{g}_1^T \boldsymbol{y} \\ \boldsymbol{g}_2^T \boldsymbol{y} \\ \boldsymbol{g}_3^T \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} \boldsymbol{g}_2^T \boldsymbol{x} \cdot \boldsymbol{g}_3^T \boldsymbol{y} - \boldsymbol{g}_3^T x \cdot \boldsymbol{g}_2^T \boldsymbol{y} \\ -\boldsymbol{g}_1^T \boldsymbol{x} \cdot \boldsymbol{g}_3^T \boldsymbol{y} + \boldsymbol{g}_3^T \boldsymbol{x} \cdot \boldsymbol{g}_1^T \boldsymbol{y} \\ \boldsymbol{g}_1^T \boldsymbol{x} \cdot \boldsymbol{g}_2^T \boldsymbol{y} - \boldsymbol{g}_2^T \boldsymbol{x} \cdot \boldsymbol{g}_1^T \boldsymbol{y} \end{bmatrix}, \tag{9}$$

where $\boldsymbol{g}_i, \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^{3\times 1}$.

Because $A^T C \cdot B^T D - A^T D \cdot B^T C = (A \times B)^T (C \times D)$, so we can have:

$$gx \times gy = \begin{bmatrix} \boldsymbol{g}_2^T \boldsymbol{x} \cdot \boldsymbol{g}_3^T \boldsymbol{y} - \boldsymbol{g}_3^T x \cdot \boldsymbol{g}_2^T \boldsymbol{y} \\ -\boldsymbol{g}_1^T \boldsymbol{x} \cdot \boldsymbol{g}_3^T \boldsymbol{y} + \boldsymbol{g}_3^T \boldsymbol{x} \cdot \boldsymbol{g}_1^T \boldsymbol{y} \\ \boldsymbol{g}_1^T \boldsymbol{x} \cdot \boldsymbol{g}_2^T \boldsymbol{y} - \boldsymbol{g}_2^T \boldsymbol{x} \cdot \boldsymbol{g}_1^T \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} (\boldsymbol{g}_2 \times \boldsymbol{g}_3)^T (\boldsymbol{x} \times \boldsymbol{y}) \\ (\boldsymbol{g}_3 \times \boldsymbol{g}_1)^T (\boldsymbol{x} \times \boldsymbol{y}) \\ (\boldsymbol{g}_1 \times \boldsymbol{g}_2)^T (\boldsymbol{x} \times \boldsymbol{y}). \end{bmatrix} \tag{10}$$

Then because:

$$\begin{aligned} &\det(g) = (\boldsymbol{g}_2 \times \boldsymbol{g}_3)^T \boldsymbol{g}_1 = \boldsymbol{g}_1^T \boldsymbol{g}_1 = 1 \\ \Longrightarrow &(\boldsymbol{g}_2 \times \boldsymbol{g}_3)^T \boldsymbol{g}_1 \boldsymbol{g}_1^{-1} = \boldsymbol{g}_1^T \boldsymbol{g}_1 \boldsymbol{g}_1^{-1} \\ \Longrightarrow &(\boldsymbol{g}_2 \times \boldsymbol{g}_3)^T = \boldsymbol{g}_1^T. \end{aligned} \tag{11}$$

Thus, we can have

$$gx \times gy = \begin{bmatrix} (\boldsymbol{g}_2 \times \boldsymbol{g}_3)^T (\boldsymbol{x} \times \boldsymbol{y}) \\ (\boldsymbol{g}_3 \times \boldsymbol{g}_1)^T (\boldsymbol{x} \times \boldsymbol{y}) \\ (\boldsymbol{g}_1 \times \boldsymbol{g}_2)^T (\boldsymbol{x} \times \boldsymbol{y}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{g}_1^T (\boldsymbol{x} \times \boldsymbol{y}) \\ \boldsymbol{g}_2^T (\boldsymbol{x} \times \boldsymbol{y}) \\ \boldsymbol{g}_3^T (\boldsymbol{x} \times \boldsymbol{y}) \end{bmatrix} = g(\boldsymbol{x} \times \boldsymbol{y}). \tag{12}$$

□

Rotation symmetric The goal is to prove

$$\text{Vector-Frame}(g\boldsymbol{x}_i, g\boldsymbol{x}_j) = g\textbf{Gram-Schmidt}\{\boldsymbol{x}_i - \boldsymbol{x}_j, \boldsymbol{x}_i - \boldsymbol{x}_k, (\boldsymbol{x}_i - \boldsymbol{x}_j) \times (\boldsymbol{x}_i - \boldsymbol{x}_k)\}. \tag{13}$$

Proof. Thus we can have:

$$\begin{aligned} \text{Vector-Frame}(g\boldsymbol{x}_i, g\boldsymbol{x}_j) &= \textbf{Gram-Schmidt}\{g\boldsymbol{x}_i - g\boldsymbol{x}_j, g\boldsymbol{x}_i - g\boldsymbol{x}_k, (g\boldsymbol{x}_i - g\boldsymbol{x}_j) \times (g\boldsymbol{x}_i - g\boldsymbol{x}_k)\} \\ &= \textbf{Gram-Schmidt}\{g(\boldsymbol{x}_i - \boldsymbol{x}_j), g(\boldsymbol{x}_i - \boldsymbol{x}_k), g((\boldsymbol{x}_i - \boldsymbol{x}_j) \times (\boldsymbol{x}_i - \boldsymbol{x}_k))\}. \end{aligned} \tag{14}$$

Recall that Gram-Schmidt projection (**Gram-Schmidt**$\{\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3\}$) is:

$$\begin{aligned} \boldsymbol{u}_1 &= \boldsymbol{v}_1, & \boldsymbol{e}_1 &= \frac{\boldsymbol{v}_1}{\|\boldsymbol{v}_1\|}, \\ \boldsymbol{u}_2 &= \boldsymbol{v}_2 - \frac{\boldsymbol{u}_1^T \boldsymbol{v}_2}{\|\boldsymbol{u}_1\|} \boldsymbol{u}_1, & \boldsymbol{e}_2 &= \frac{\boldsymbol{v}_2}{\|\boldsymbol{v}_2\|}, \\ \boldsymbol{u}_3 &= \boldsymbol{v}_3 - \frac{\boldsymbol{u}_1^T \boldsymbol{v}_3}{\|\boldsymbol{u}_1\|} \boldsymbol{u}_1 - \frac{\boldsymbol{u}_2^T \boldsymbol{v}_3}{\|\boldsymbol{u}_2\|} \boldsymbol{u}_2, & \boldsymbol{e}_3 &= \frac{\boldsymbol{v}_3}{\|\boldsymbol{v}_3\|}. \end{aligned} \tag{15}$$

Thus, the Gram-Schmidt projection on the rotated vector (**Gram-Schmidt**$\{g\boldsymbol{v}_1, g\boldsymbol{v}_2, g\boldsymbol{v}_3\}$) is:

$$\begin{aligned} \boldsymbol{u}_1' &= g\boldsymbol{v}_1, \\ \boldsymbol{u}_2' &= g\boldsymbol{v}_2 - g\frac{\boldsymbol{u}_1^T \boldsymbol{v}_2}{\|\boldsymbol{u}_1\|} \boldsymbol{u}_1, \\ \boldsymbol{u}_3' &= g\boldsymbol{v}_3 - g\frac{\boldsymbol{u}_1^T \boldsymbol{v}_3}{\|\boldsymbol{u}_1\|} \boldsymbol{u}_1 - g\frac{\boldsymbol{u}_2^T \boldsymbol{v}_3}{\|\boldsymbol{u}_2\|} \boldsymbol{u}_2, \end{aligned} \tag{16}$$

Thus, **Gram-Schmidt**$\{g\boldsymbol{v}_1, g\boldsymbol{v}_2, g\boldsymbol{v}_3\} = g$**Gram-Schmidt**$\{\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3\}$.

□

Transition symmetric

$$\text{Vector-Frame}(\boldsymbol{x}_i + \delta\boldsymbol{x}, \boldsymbol{x}_j + \delta\boldsymbol{x}) = \textbf{Gram-Schmidt}\{\boldsymbol{x}_i - \boldsymbol{x}_j, \boldsymbol{x}_i - \boldsymbol{x}_k, (\boldsymbol{x}_i - \boldsymbol{x}_j) \times (\boldsymbol{x}_i - \boldsymbol{x}_k)\}. \tag{17}$$

Proof. Because the basis is based on the difference of coordinates, it is straightforward to observe that **Gram-Schmidt**$\{\boldsymbol{v}_1 + \boldsymbol{t}, \boldsymbol{v}_2 + \boldsymbol{t}, \boldsymbol{v}_3 + \boldsymbol{t}\} =$ **Gram-Schmidt**$\{\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3\}$. So the frame operation is transition equivariant. □

Reflection antisymmetric

$$\text{Vector-Frame}(\boldsymbol{x}_i, \boldsymbol{x}_j) \neq \text{Vector-Frame}(-\boldsymbol{x}_i, -\boldsymbol{x}_j). \tag{18}$$

Proof. From intuition, this makes sense because the cross-product is anti-symmetric.

A simple counter-example is the original geometry R and the reflected geometry by the original point $-R$. Thus the two bases before and after the reflection group is the following:

$$\textbf{Gram-Schmidt}\{\boldsymbol{x}_i - \boldsymbol{x}_j, \boldsymbol{x}_i - \boldsymbol{x}_k, (\boldsymbol{x}_i - \boldsymbol{x}_j) \times (\boldsymbol{x}_i - \boldsymbol{x}_k)\} \tag{19}$$

$$\textbf{Gram-Schmidt}\{-\boldsymbol{x}_i + \boldsymbol{x}_j, -\boldsymbol{x}_i + \boldsymbol{x}_k, (\boldsymbol{x}_i - \boldsymbol{x}_j) \times (\boldsymbol{x}_i - \boldsymbol{x}_k)\}. \tag{20}$$

The bases between $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3$ and $\{-\boldsymbol{v}_1, -\boldsymbol{v}_2, \boldsymbol{v}_3\}\}$ are different, thus such frame construction is reflection anti-symmetric.

□

D.2 Scalarization

Once we have the three vectors as the vector frame basis, the next thing is to do modeling. Suppose the frame is $\mathcal{F} = (\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3)$, then for an equivariant vector (tensor) $\boldsymbol{h}$, the scalarization is:

$$\boldsymbol{h} \odot \mathcal{F} = (\boldsymbol{h} \odot \boldsymbol{e}_1, \boldsymbol{h} \odot \boldsymbol{e}_2, \boldsymbol{h} \odot \boldsymbol{e}_3) = (\boldsymbol{h}_1, \boldsymbol{h}_2, \boldsymbol{h}_3). \tag{21}$$

D.3 Multi-Grained SE(3)-Equivariant Vector Frame

Proteins are essentially macromolecules composed of thousands of residues (amino acids), where each residue is a small molecule. Thus, it is infeasible to model all the atoms in proteins due to the large volume of the system, and such an issue also holds for the protein-ligand complex. To address this issue, we propose BindingNet, a multi-grained SE(3)-equivariant model. The vector frame basis ensures SE(3)-equivariance, and the multi-granularity is achieved by considering frames at three levels.

Vector Frame Basis for SE(3)-Equivariance. Recall that the geometric representation of the whole molecular system needs to follow the physical properties of the equivariance w.r.t. rotation and translation. Such a group symmetric property is called SE(3)-equivariance. We also want to point out that the reflection or chirality property is equivariant for properties like energy, yet it is not for the ligand modeling with rigid protein structures (*i.e.*, antisymmetric to the reflection). The vector frame basis can handle this naturally, and we leave a more detailed discussion in Appendix D, along with the proof on group symmetry of vector frame basis. In the following, we introduce three levels of vector frames for multi-grained modeling.

Atom-Level Vector Frame for Ligands. For small molecule ligands, we first extract atom pairs (i, j) within the distance cutoff c, and the vector frame basis is constructed using the Gram-Schmidt as:

$$\mathcal{F}_{\text{ligand}} = \left(\frac{\boldsymbol{x}_i^{(l)} - \boldsymbol{x}_j^{(l)}}{\left\|\boldsymbol{x}_i^{(l)} - \boldsymbol{x}_j^{(l)}\right\|}, \frac{\boldsymbol{x}_i^{(l)} \times \boldsymbol{x}_j^{(l)}}{\left\|\boldsymbol{x}_i^{(l)} \times \boldsymbol{x}_j^{(l)}\right\|}, \frac{\boldsymbol{x}_i^{(l)} - \boldsymbol{x}_j^{(l)}}{\left\|\boldsymbol{x}_i^{(l)} - \boldsymbol{x}_j^{(l)}\right\|} \times \frac{\boldsymbol{x}_i^{(l)} \times \boldsymbol{x}_j^{(l)}}{\left\|\boldsymbol{x}_i^{(l)} \times \boldsymbol{x}_j^{(l)}\right\|}\right), \tag{22}$$

where $\times$ is the cross product. Note that both $\boldsymbol{x}_i^{(l)}$ and $\boldsymbol{x}_j^{(l)}$ are for geometries at time t - henceforth, we omit the subscript t for brevity. Such an atom-level vector frame allows us to do SE(3)-equivariant message passing to get the atom-level representation.

Backbone-Level Vector Frame for Proteins. Proteins can be treated as chains of residues, where each residue possesses a backbone structure. The backbone structure comprises an amino group, a carboxyl group, and an alpha carbon, delegated as $N - C_\alpha - C$. Such a structure serves as a natural way to build the vector frame. For each residue in the protein, the coordinates are $\boldsymbol{x}_N$, $\boldsymbol{x}_{C_\alpha}$, and $\boldsymbol{x}_C$, then the backbone-level vector frame for this residue is:

$$\mathcal{F}_{\text{protein}} = \left(\frac{\boldsymbol{x}_N - \boldsymbol{x}_{C_\alpha}}{\|\boldsymbol{x}_N - \boldsymbol{x}_{C_\alpha}\|}, \frac{\boldsymbol{x}_{C_\alpha} - \boldsymbol{x}_C}{\|\boldsymbol{x}_{C_\alpha} - \boldsymbol{x}_C\|}, \frac{\boldsymbol{x}_N - \boldsymbol{x}_{C_\alpha}}{\|\boldsymbol{x}_N - \boldsymbol{x}_{C_\alpha}\|} \times \frac{\boldsymbol{x}_{C_\alpha} - \boldsymbol{x}_C}{\|\boldsymbol{x}_{C_\alpha} - \boldsymbol{x}_C\|}\right). \tag{23}$$

This is built for each residue, providing a residue-level representation.

Residue-Level Vector Frame for Protein-Ligand Complexes. It is essential to model the protein-ligand interaction to better capture the binding dynamics. We achieve this by introducing the residue-level vector frame.

More concretely, proteins are sequences of residues, marked as $\{(f_0^{(p)}, \boldsymbol{x}_0^{(p)}), ..., (f_i^{(p)}, \boldsymbol{x}_i^{(p)}), (f_{i+1}^{(p)}, \boldsymbol{x}_{i+1}^{(p)}, ...\}$. Here, we use a cutoff threshold c to determine the interactions between ligands and proteins, and the interactive regions on proteins are called pockets. We construct the following vector frame for residues in the pockets sequentially:

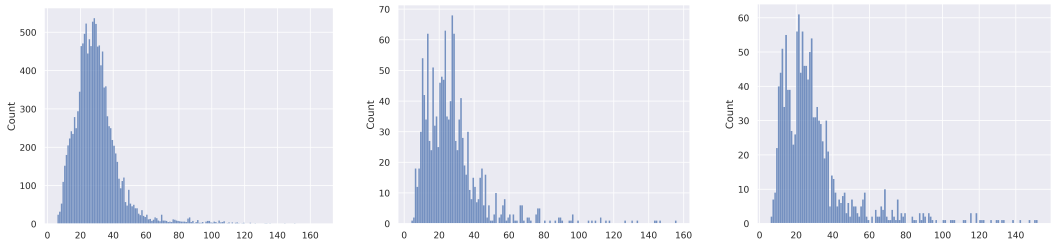
$$\mathcal{F}_{\text{complex}} = \left(\frac{\boldsymbol{x}_i^{(p)} - \boldsymbol{x}_{i+1}^{(p)}}{\left\|\boldsymbol{x}_i^{(p)} - \boldsymbol{x}_{i+1}^{(p)}\right\|}, \frac{\boldsymbol{x}_i^{(p)} \times \boldsymbol{x}_{i+1}^{(p)}}{\left\|\boldsymbol{x}_i^{(p)} \times \boldsymbol{x}_{i+1}^{(p)}\right\|}, \frac{\boldsymbol{x}_i^{(p)} - \boldsymbol{x}_{i+1}^{(p)}}{\left\|\boldsymbol{x}_i^{(p)} - \boldsymbol{x}_{i+1}^{(p)}\right\|} \times \frac{\boldsymbol{x}_i^{(p)} \times \boldsymbol{x}_{i+1}^{(p)}}{\left\|\boldsymbol{x}_i^{(p)} \times \boldsymbol{x}_{i+1}^{(p)}\right\|}\right). \tag{24}$$

Through this complex-level vector frame, the message passing enables the exchange of information between atoms from ligands and residues from the pockets. The illustration of the above three levels of vector frames can be found in Figure 4. Once we build up such three vector frames, we then conduct a *scalarization* operation [7], which transforms the equivariant variables (*e.g.*, coordinates) to invariant variables by projecting them to the three vector bases in the vector frame.

E Specifications on MISATO

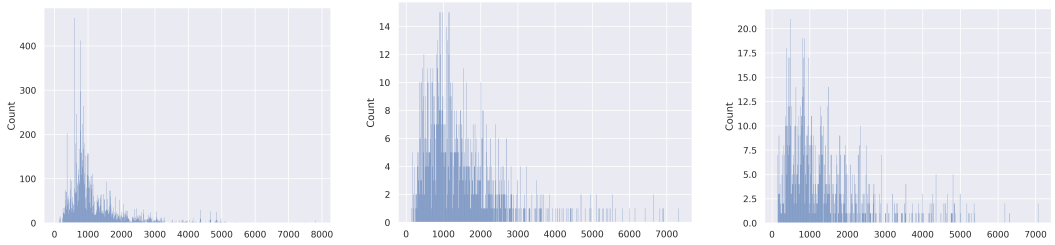
In this section, we provide more details on the MISATO dataset [35].

For small molecule ligands, we ignore the Hydrogen atoms.



(a) Training data. (b) Validation data. (c) Test data.

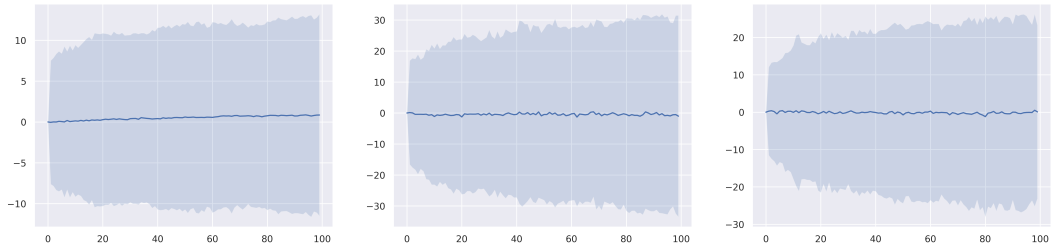
Figure 5: Distribution on # atoms in small molecule ligands for all protein-ligand complex.



(a) Training data. (b) Validation data. (c) Test data.

Figure 6: Distribution on # residues in proteins for all protein-ligand complex.

We also plot the distribution of the energy gap between each time step and the initial snapshot, *i.e.*, $E_t - E_0$. The distribution is in Figure 7. We can observe that as the time processes, the mean of the energy stays almost the same, yet the variance gets higher.



(a) Training data. (b) Validation data. (c) Test data.

Figure 7: Distribution on energy $E_t - E_0$.

F Details of NeuralMD

Residue-Level Complex Modeling. Once we obtain the atom-level representation and vector ($\boldsymbol{h}^{(l)}, \text{vec}^{(l)}$) from ligands, and backbone-level representation ($\boldsymbol{h}^{(p)}$) from proteins, the next step is to learn the protein-ligand interaction. We first extract the residue-atom pair (i, j) with a cutoff c, based on which we obtain an equivariant interaction edge representation $\boldsymbol{h}_{ij} = (\boldsymbol{h}_i^{(l)} + \boldsymbol{h}_j^{(p)}) \cdot (\boldsymbol{x}_i^{(l)} - \boldsymbol{x}_j^{(p)})$. After scalarization, we can obtain invariant interaction edge representation $\boldsymbol{h}_{ij} = \boldsymbol{h}_{ij} \cdot \mathcal{F}_{\text{complex}}$. Finally, we adopt an equivariant MPNN layer to get the atom-level force as:

$$\text{vec}_{ij}^{(pl)} = \text{vec}_i^{(l)} \cdot \text{MLP}(h_{ij}) + (x_i^{(l)} - x_j^{(p)}) \cdot \text{MLP}(h_{ij}), \quad F_i^{(l)} = \text{vec}_i^{(l)} + \text{Agg}_{j \in \mathcal{N}(i)} \text{vec}_{ij}^{(pl)}. \tag{25}$$

In the last equation, the ultimate force predictions can be viewed as two parts: the internal force from the molecule $\text{vec}_i^{(l)}$ and the external force from the protein-ligand interaction $\text{vec}_{ij}^{(pl)}$.

F.1 Architecture details

In this section, we provide more details on the model architecture in Figure 8, and hyperparameter details in Table 3.

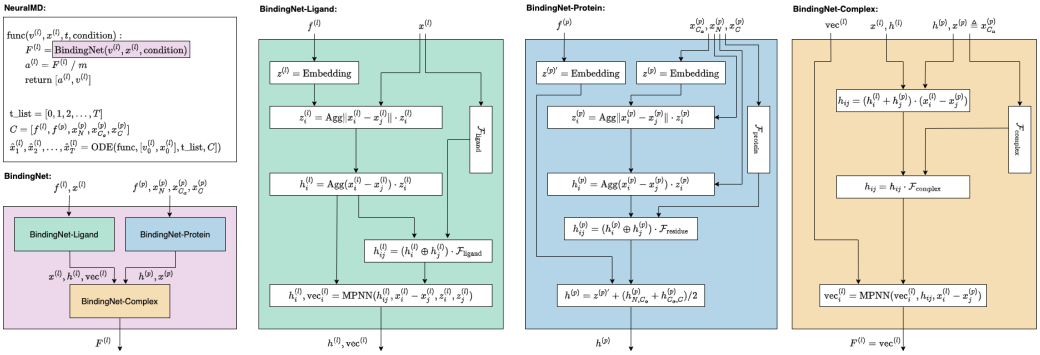


Figure 8: Detailed pipeline of NeuralMD. In the three key modules of BindingNet, there are three vertical boxes, corresponding to three granularities of vector frames, as in Equations (22) to (24).

Table 3: Hyperparameter specifications for NeuralMD.

Hyperparameter	Value
# layers	{5}
c	{5}
cutoff c	5
learning rate	{1e-4, 1e-3}
optimizer	{SGD, Adam }

G More Experiment Results

G.1 Experiment Setup

To verify the effectiveness and efficiency of NeuralMD, we design ten single-trajectory and three multi-trajectory binding simulation tasks. For evaluation, we adopt the recovery and stability metrics [11]. NeuralMD achieves 2000× speedup compared to the numerical methods. We observe that NeuralMD outperforms all other ML methods [47, 27, 12, 44, 1] on 12 tasks using recovery metric, and NeuralMD is consistently better by a large gap using the stability metric (up to ~80%). Qualitatively, we illustrate that NeuralMD realizes more stable binding dynamics predictions in three case studies. They are three protein-ligand binding complexes from Protein Data Bank (PDB), as shown in Figure 1.

Baselines. In this work, we mainly focus on machine learning methods for trajectory prediction, *i.e.*, no energy or force labels are considered. GNN-MD is to apply geometric graph neural networks (GNNs) to predict the trajectories in an auto-regressive manner [35, 12]. More concretely, GNN-MD takes as inputs the geometries at time t and predicts the geometries at time $t + 1$. DenoisingLD (denoising diffusion for Langevin dynamics) [1, 44, 12] is a baseline method that models the trajectory prediction as denoising diffusion task [37], and the inference for trajectory generation essentially becomes the Langevin dynamics. CG-MD learns a dynamic GNN and a score GNN [12], which are essentially the hybrid of GNN-MD and DenoisingLD. Here, to make the comparison more explicit, we compare these two methods (GNN-MD and DenoisingLD) separately. Additionally, we consider VerletMD, an energy prediction research line (including DeePMD [47], TorchMD [6], and Allegro-LAMMPS [27]), where the role of ML models is to predict the energy, and the MD trajectory is obtained by the velocity Verlet algorithm, a numerical integration method for Newtonian mechanics. We keep the same backbone model (BindingNet) for energy or force prediction for all the baselines.

G.2 More results on single-trajectory prediction

One main benefit of using NeuralMD for binding simulation is its efficiency. To show this, we list the computational time in Table 4. We further approximate the wall time of the numerical method for MD simulation (PDB 5WIJ). Concretely, we can get an estimated speed of 1 nanosecond of dynamics every 0.28 hours. This is running the simulation with GROMACS [42] on 1 GPU with 16 CPU cores and a moderately sized water box at the all-atom level (with 2 femtosecond timesteps). This equivalently shows that NeuralMD is ~2000× faster than numerical methods.

Table 4: Efficiency comparison of FPS between VerletMD and NeuralMD on single-trajectory prediction.

PDB ID	5WIJ	4ZX0	3EOV	4K6W	1KTI	1XP6	4YUR	4G3E	6B7F	3B9S	Average
VerletMD	12.564	30.320	29.890	26.011	19.812	28.023	31.513	29.557	19.442	31.182	25.831
NeuralMD (Ours)	33.164	39.415	31.720	31.909	24.566	37.135	39.365	39.172	20.320	37.202	33.397

G.3 MD Prediction: Generalization Among Multiple Trajectories

A more challenging task is to test the generalization ability of NeuralMD among different trajectories. The MISATO dataset includes 13,765 protein-ligand complexes, and we first create two small datasets by randomly sampling 100 and 1k complexes, respectively. Then, we take 80%-10%-10% for training, validation, and testing. We also consider the whole MISATO dataset, where the data split has already been provided. After removing the peptide ligands, we have 13,066, 1,357, and 1,357 complexes for training, validation, and testing, respectively.

The quantitative results are in Table 5. First, we can observe that VerletMD has worse performance on all three datasets, and the performance gap with other methods is even larger compared to the single-trajectory prediction. The other two baselines, GNN-MD and DenoisingLD, show similar performance, while NeuralMD outperforms in all datasets. Notice that stability (%) remains more distinguishable than the two trajectory recovery metrics (MAE and MSE).

Table 5: Results on three multi-trajectory binding dynamics predictions. Results with optimal validation loss are reported. Four evaluation metrics are considered: MAE (Å, ↓), MSE (↓), and Stability (%, ↓).

Dataset	MISATO-100			MISATO-1000			MISATO-All		
	MAE	MSE	Stability	MAE	MSE	Stability	MAE	MSE	Stability
VerletMD	90.326	56.913	86.642	80.187	53.110	86.702	105.979	69.987	90.665
GNN-MD	7.176	4.726	35.431	7.787	5.118	33.926	8.260	5.456	32.638
DenoisingLD	7.112	4.684	29.956	7.746	5.090	18.898	15.878	10.544	89.586
NeuralMD (Ours)	**6.852**	**4.503**	**19.173**	**7.653**	**5.028**	**15.572**	**8.147**	**5.386**	**17.468**

G.4 Ablation Studies: Flexible Binding

Recall that, in the main paper, we have discussed using the *semi-flexible* binding setting, *i.e.*, proteins with rigid structures while small molecule ligands with flexible structures, and the goal is to predict the trajectories of the ligands. If we want to take both proteins and ligands with flexible structures, one limitation is the GPU memory cost. However, we would like to mention that it is possible to do NeuralMD on protein-ligand with small volume, and we take an ablation study to test them as below.

Problem Setup. Both the proteins and ligands are flexible, and we want to predict their trajectories simultaneously. In the main paper, we consider three levels of vector frames. Here in the flexible setting, due to the large volume of atoms in the protein-ligand complex, we are only able to consider two levels, *i.e.*, the atom-level and residue-level. Thus, the backbone model (BindingNet) also changes accordingly. The performance is shown in Table 6, and we can see that NeuralMD is consistently better than the GNN-MD on all three metrics and all 10 single trajectories. We omit the multi-trajectory experiments due to the memory limitation.

Table 6: Results on ten single-trajectory binding dynamics prediction. Results with optimal training loss are reported. Four evaluation metrics are considered: MAE (Å, ↓), MSE (↓), and Stability (%, ↓).

		GNN-MD	NeuralMD (Ours)
5WIJ	MAE	7.126	**3.101**
	MSE	4.992	**2.070**
	Stability	68.317	**30.655**
4ZX0	MAE	9.419	**2.580**
	MSE	6.269	**1.724**
	Stability	67.492	**29.013**
3EOV	MAE	10.695	**3.664**
	MSE	7.447	**2.521**
	Stability	67.782	**39.714**
4K6W	MAE	8.347	**3.056**
	MSE	5.605	**2.007**
	Stability	63.839	**36.972**
1KTI	MAE	8.900	**6.815**
	MSE	5.820	**4.268**
	Stability	65.010	**26.805**
1XP6	MAE	8.496	**1.910**
	MSE	5.673	**1.276**
	Stability	70.019	**33.907**
4YUR	MAE	11.710	**7.568**
	MSE	7.759	**4.966**
	Stability	69.163	**34.636**
4G3E	MAE	1314.425	**3.282**
	MSE	814.641	**2.152**
	Stability	65.703	**21.095**
6B7F	MAE	182.278	**3.166**
	MSE	115.688	**2.121**
	Stability	72.027	**26.931**
3B9S	MAE	3.590	**2.477**
	MSE	2.431	**1.615**
	Stability	54.890	**18.817**