Using pretrained graph neural networks with token mixers as geometric featurizers for conformational dynamics

Zihan Pengmei,¹ Chatipat Lorpaiboon,¹ Spencer C. Guo,¹ Jonathan Weare,² and Aaron R. Dinner^{1, a)} ¹⁾Department of Chemistry and James Franck Institute, University of Chicago, Chicago, Illinois 60637, United States

²⁾Courant Institute of Mathematical Sciences, New York University, New York, New York 10012, United States

Identifying informative low-dimensional features that characterize dynamics in molecular simulations remains a challenge, often requiring extensive manual tuning and system-specific knowledge. Here, we introduce geom2vec, in which pretrained graph neural networks (GNNs) are used as universal geometric featurizers. By pretraining equivariant GNNs on a large dataset of molecular conformations with a self-supervised denoising objective, we obtain transferable structural representations that are useful for learning conformational dynamics without further fine-tuning. We show how the learned GNN representations can capture interpretable relationships between structural units (tokens) by combining them with expressive token mixers. Importantly, decoupling training the GNNs from training for downstream tasks enables analysis of larger molecular graphs (such as small proteins at all-atom resolution) with limited computational resources. In these ways, geom2vec eliminates the need for manual feature selection and increases the robustness of simulation analyses.

I. INTRODUCTION

Molecular dynamics simulations can provide atomistic insight into complex reaction dynamics, but their high dimensionality makes them hard to interpret. Analyzing simulations thus relies on identifying low-dimensional representations (features), but care is needed in choosing them because they can strongly impact conclusions 1-4. Often features are selected manually, but doing so relies on system-specific intuition. Because developing intuition is often the goal of simulations, many researchers instead have turned to machine-learning methods for constructing features that capture observed variance (e.g., principal component analysis)⁵⁻⁷ or decorrelate slowly according to the simulation data (e.g., the variational approach for Markov processes, VAMP)⁸⁻¹². While these objectives are not always aligned with the reaction of interest^{3,4,13,14}, such features can serve as useful intermediaries for computing reaction-specific statistics^{14,15} that provide a principled way for evaluating mechanisms $^{16-18}$.

Generally the inputs to the machine-learning methods above are internal coordinates such as distances between selected atoms and dihedral angles because they are invariant to translations and rotations of the system. However, the nonlocal nature of these coordinates and/or their effects (e.g., the rotation of a dihedral angle in a polymer backbone) can make the resulting features both ineffective and hard to interpret, and these issues become more significant with system size^{16,17}. Additionally, it is not obvious how to represent permutationally invariant species such as solvent molecules with internal coordinates; recently introduced machine learning approaches for treating such species do not scale well with system size¹⁹.

Because atoms and their interactions (through bond or through space) can be viewed as the nodes and edges of graphs, molecular information can be readily encoded in graph representations (e.g., graph neural networks, GNNs). Importantly, graph representations can be constructed in ways that respect the symmetries of molecular systems, with translational, rotational, and permutational invariance and equivariance. Equivariance allows, for example, GNNs to output forces that rotate with the system, and appears to improve learning $^{20-22}$. Owing to both their conceptual appeal and their performance, GNNs now dominate machine learning for force fields and molecular property prediction 22-26. They also are being successfully used to learn representations of larger molecules for tasks such as protein structure prediction, design, fold classification, and function prediction $^{27-30}$.

The tasks above concern prediction of static properties, including structures. Because molecular dynamics trajectories consist of sequences of structures, GNNs should be useful for identifying features for computing reaction statistics, and several groups have combined GNNs with $VAMP^8$ to learn metastable states and relaxation time scales of both materials and biomolecules $^{31-35}$. These groups report improved variational scores, convergence for shorter lag times, and more interpretable learned representations relative to VAMPnets based on fully connected networks. However, existing GNNs for analyzing dynamics do not readily scale to large numbers of atoms, so the graphs in these studies are small, either because the molecules are small, or only a subset of atoms (e.g., the C_{α} atoms of proteins) are used as inputs. Furthermore, training these GNNs is computationally costly, limiting the number of architectures that can be explored and their use for other types of analyses.

The key idea of this paper is that GNNs can be pretrained using independent structural data prior to their use to analyze dynamics, thus decoupling GNN training from training for downstream tasks. Pretraining

^{a)}Electronic mail: dinner@uchicago.edu

2

A. Network architecture

As noted above, our goal is learn a mapping from Cartesian coordinates to representation vectors via a GNN. There are many existing GNN architectures from which to $choose^{23}$. Here, we use the ViSNet⁵³ architecture, which is built on $TorchMD-ET^{54}$. These are both equivariant geometric graph transformers with modified attention mechanisms that suppress interactions between distant atoms; ViSNet goes beyond TorchMD-ET in using (standard and improper) dihedral angle information in its internal representations. We take the activations after the last message-passing layer as the representations for downstream computations. These representations include three-dimensional vector features, which change appropriately with molecular translation and rotation, and one-dimensional scalar features, which are invariant to molecular translation and rotation. That is,

$$f_{\text{GNN}} : \mathbb{R}^{3N} \to \mathbb{R}^{(1+3)dN}, \tag{1}$$

where N is the number of atoms, and d is the number of features associated with each atom (equal to the dimension of the last update layer). We represent the combined vector ($\mathbf{v}_i \in \mathbb{R}^{3d}$) and scalar ($x_i \in \mathbb{R}^d$) features for atom i by $\mathbf{h}_i \in \mathbb{R}^{(1+3)d}$. We choose ViSNet because it was previously shown to give accurate molecular property predictions and accurate conformational distributions when used to learn a potential for molecular dynamics simulations.⁵³ We briefly introduce the TorchMD-ET and ViSNet architectures in Appendices A 2 and A 3, and we refer interested readers to the original publications and Ref. 55 for further details.

B. Pretraining by denoising

For the pretraining, we draw random displacements from a multivariate Gaussian distribution and add them to the Cartesian coordinates of the molecules in the training set; we then train the GNN to predict the displacements. This process is designed to encourage the network to learn representations that capture the geometry of the molecular conformations, and it can be viewed as learning a force field with energy minima close to the training set geometries⁴⁰. We choose this objective because structural data are more readily available than energetic data, especially for large molecules such as proteins.

Here, we use the OrbNet Denali dataset, which consists of 215,000 molecules and complexes (with an average of 45 atoms) with 2.3 million conformations sampled from molecular trajectories⁴⁴. We randomly select 10,000 conformations for validation and use the remainder for training.

Following Zaidi *et al.*⁴⁰, we pretrain the model by passing the (1 + 3)d features for each atom (graph node) from the ViSNet architecture through a gated equivariant block (Algorithm 1) that combines the *d* scalar and

has transformed other domains such as natural language processing (NLP) and computer vision, enabling highdimensional latent representations of words ("tokens")³⁶ or images ("patches")³⁷ to be learned by self-supervised, auto-regressive training. In NLP, word2vec pioneered the idea of using learnable vector representations for words by assigning them based on the word itself and its surrounding context³⁸; these representations could then be used for diverse downstream tasks. Inspired by word2vec, we propose geom2vec, an approach that leverages pretrained GNNs to learn transferable vector representations for molecular geometries.

Various pretraining strategies have been tried in molecular contexts^{27,28,30,39–41}, but in contrast to complex pretraining and encoding schemes devised for specific classes of molecules^{27,30}, we use a scheme that can be applied generally. Building on the idea that corrupting data with noise and training a model to reconstruct the original data (denoising) can lead to learning meaningful representations for generative models^{42,43}, Zaidi *et al.*⁴⁰ showed that denoising atomic coordinates of structures of organic molecules significantly improved GNN performance on a number of molecular property prediction benchmarks.

Here, we show that this simple pretraining scheme also enables analysis of molecular dynamics simulations. Specifically, we pretrain GNNs using the same denoising objective and a dataset of structures of organic molecules obtained with density functional theory⁴⁴ and analyze protein molecular dynamics simulations with the resulting representations. We consider two tasks: learning slowly decorrelating modes with VAMP⁸ and identifying metastable states with the state predictive information bottleneck (SPIB) framework^{45,46}. We show that neural networks trained using the GNN representations can readily take all non-hydrogen atoms in a small protein as inputs, enabling, for example, discovery of side chain dynamics that are important for folding. By decoupling learning molecular representations from training for specific tasks, our method naturally accommodates alternative pretraining schemes^{27,47} and datasets⁴⁸ (e.g., ones specific to particular classes of molecules), as well as other possible tasks $^{49-52}$.

II. METHODS

The basic idea of our method is to pretrain a GNN using a suitable task (here, denoising molecular coordinates) and then to use it with the resulting parameter values fixed as a feature encoder for other (downstream) tasks, as summarized in Figure 1. In this section, we provide an overview of the network architecture that we use and then describe its elaboration for the pretraining and downstream tasks; further details are provided in the Appendices. We refer to the workflow of transforming the atomic coordinates to representation vectors (i.e., features) and the use of those vectors as "geom2vec."



FIG. 1. The geom2vec workflow. (a) A GNN encoder is pretrained using a denoising objective on a dataset of structures of diverse molecules. (b) Geometric representations for configurations from molecular simulations are obtained by performing inference with the pretrained GNN encoder. (c) The representations are used as inputs to a downstream task head (here, a VAMPnet or SPIB), which is trained separately.

vector features to obtain a three-dimensional vector that represents the predictions for the displacement of that atom. We train for a fixed number of epochs and save the parameters resulting in the lowest mean squared error (MSE) between the predicted and added atom displacements for the validation set; for this comparison, we normalize the added atom displacements to have zero mean and unit variance. Further details, including hyperparameters, are given in Appendix D. Depending on the network architecture, choice of hyperparameters, and graphics card available, the one-time pretraining can take from several hours to a few days.

C. Use of the representations

In this section, we discuss the operational details of using the pretrained GNN for downstream tasks in general (summarized in Figure 1). The specific downstream tasks that we use for our numerical demonstrations are described in Section III.

1. Atom selection

We first select the atoms of interest. For example, when analyzing simulations with explicit solvent, we may select only the solute coordinates. Given the molecular coordinates, the pretrained GNN yields a learned representation \mathbf{h}_i for each selected atom *i*. One can choose to base the calculations for the downstream tasks on the atomic representation vectors and/or the sums over their *d* scalar and vector elements, but in most cases we reduce the size of the graph by coarse-graining it.

2. Coarse-graining

Let $S = \{S_1, \ldots, S_M\}$ be a partition of the atoms into M disjoint subsets, where each subset S_m contains atoms belonging to a structural unit, such as a functional group or monomer in a polymer, depending on the system. We pool the atomic representations for each S_m :

$$\bar{\mathbf{h}}_{S_m} = \sum_{i \in S_m} \mathbf{h}_i. \tag{2}$$

Each resulting coarse-grained representation vector $\mathbf{\bar{h}}_{S_m}$ represents the geometric information of the atoms within its structural unit in an average sense. Following the NLP literature, we refer to the coarse-grained representations as structural "tokens."

How to partition the atoms is the user's choice, but many systems have a natural structure. For example, here we study proteins and pool the representations for the atoms in each amino acid. In this case, the number of nodes in the graph is reduced by an order of magnitude. Coarse-graining reduces the computational cost and memory, and it facilitates both learning long-range relationships and interpretation of the results (e.g., attention maps).

3. Feature combination

To use the coarse-grained tokens in a learning task, we must combine their information in a useful fashion. How best to do so depends on the structural properties of interest, but we can generally lump approaches into two categories:

- Pooling: We sum (or average) the coarse-grained tokens to obtain a single, global representation of the molecular system $\bar{\mathbf{h}} \in \mathbb{R}^{(1+3)d}$. Note that this approach is equivalent to pooling the atomic representation vectors \mathbf{h}_i .
- Token mixing: This approach applies a learnable mixing operation to the coarse-grained tokens to capture their interactions and dependencies⁵⁶. Token mixers allow for greater expressivity than direct pooling and can capture complex interactions between structural units.

In this work, we employ two basic token mixers: (1) a standard transformer architecture³⁶, which we refer to as "SubFormer,"⁵⁷ and (2) an MLP-mixer architecture⁵⁸, which we refer to as "SubMixer." Because the GNNs here employ a message passing architecture (Appendix A), the resulting features typically do not encode global geometric information. We show that this issue can be addressed by combining SubFormer and SubMixer with a special token that encodes global information⁵⁹ such as pairwise distances. Alternatively, they can be combined with geometric vector perceptrons (GVPs), equivariant GNNs introduced for biomolecular modeling⁶⁰ to learn expressive positional encodings (see Algorithms 2 and 3 in Appendices B and C).

D. Output layers

Ultimately, we combine the features from the different graph nodes and any graph-wide information (e.g., the CLS token of the transformer) and use an MLP to output quantities specific to a downstream task. Here, we consider downstream tasks that require only scalar quantities, so we input only the scalar features to the MLP, but generally scalar (invariant) and vector (equivariant) quantities can be input and output. We summarize the overall scheme in Algorithm 3.

III. DOWNSTREAM TASKS

To assess whether the geom2vec representations are useful for learning protein dynamics, we apply them to learning slowly decorrelating modes with VAMP⁸ and identifying metastable states with the state predictive information bottleneck (SPIB) framework⁴⁵. As described previously, we apply the pretrained GNN to the coordinates from molecular dynamics trajectories and then use the resulting features as inputs to the desired task without further fine-tuning the GNN parameters. In this section, we briefly describe the two downstream tasks mathematically.

A. VAMPnets

Let \mathbf{X}_t be a Markov process and define the correlation functions

$$C_{00} = \mathbf{E}_{\mathbf{X}_0 \sim \mu} [\chi_0(\mathbf{\hat{h}}(\mathbf{X}_0))\chi_0^{\mathsf{T}}(\mathbf{\hat{h}}(\mathbf{X}_0))]$$
(3)

$$C_{0\tau} = \mathbf{E}_{\mathbf{X}_0 \sim \mu} [\chi_0(\tilde{\mathbf{h}}(\mathbf{X}_0)) \chi_{\tau}^{\mathsf{T}}(\tilde{\mathbf{h}}(\mathbf{X}_{\tau}))]$$
(4)

$$C_{\tau\tau} = \mathbf{E}_{\mathbf{X}_0 \sim \mu} [\chi_{\tau}(\tilde{\mathbf{h}}(\mathbf{X}_{\tau})) \chi_{\tau}^{\mathsf{T}}(\tilde{\mathbf{h}}(\mathbf{X}_{\tau}))], \qquad (5)$$

where χ_0 and χ_{τ} are vectors of functions and the expectation is over trajectories initialized from an arbitrary distribution μ . In our case, \mathbf{X}_t represents the molecular coordinates at time t and $\tilde{\mathbf{h}}(\mathbf{X})$ are the molecular features from a pretrained GNN. The variational approach for Markov processes (VAMP)⁸⁻¹² states that χ_0 and χ_{τ} represent the slowest decorrelating modes (or collective variables; CVs) of the system when maximizing the VAMP-2 score

VAMP-2 =
$$\left\| C_{00}^{-1/2} C_{0\tau} C_{\tau\tau}^{-1/2} \right\|_{\mathrm{F}}^{2}$$
, (6)

where the subscript F denotes the Frobenius norm.

Operationally, the components of χ_0 and χ_{τ} are learned from data by representing them by parameterized functions (e.g., neural networks in VAMPnets^{8,10}; the output of geom2vec in our case) and maximizing (6). VAMPnets require one to specify the output dimension d_o a priori. For the benchmark systems that we consider, we choose d_o based on previous results in the literature.

As discussed below (Section IV D) we split each dataset into training and validation sets, evaluating the validation score every 10 training steps. Each training step or validation step, we randomly draw a batch of trajectoryframe pairs spaced by τ to compute (6). We found that a large batch size (at least 1000 and usually 5000 for the examples here) was required to achieve a high validation score. With smaller batch sizes, we encountered numerical instabilities when inverting the correlation matrices in the VAMP-2 loss function (6). A large batch size is also needed to minimize variance in the late phase of training because neural network outputs exhibit large changes between metastable states, where fewer trajectory-frame pairs contribute. To prevent overfitting, we apply an early stopping criterion¹²: we stop training when the training VAMP score does not increase for 500 batches or the validation VAMP score does not increase for 10 batches. Further training details are given in Table S3.

B. State Predictive Information Bottleneck (SPIB)

In the information bottleneck (IB) framework, an encoder-decoder setup is used to learn a low-dimensional (latent) representation \mathbf{z} that minimizes the information from a high-dimensional input \mathbf{x} while maximizing the information about a target \mathbf{y} . The associated loss func-

tion is

$$\mathcal{L}_{\rm IB} = I(\mathbf{z}, \mathbf{y}) - \beta I(\mathbf{x}, \mathbf{z}), \tag{7}$$

where I refers to the mutual information between two random variables:

$$I(A,B) = \int p(A,B) \ln \frac{p(A,B)}{p(A)p(B)} dA dB, \qquad (8)$$

and the parameter β controls the tradeoff between prediction accuracy and the complexity of the latent representation.

In the state predictive information bottleneck (SPIB) extension of IB⁴⁵, the inputs are molecular features at time t, $\tilde{\mathbf{h}}(\mathbf{X}_t)$, and the targets are state labels s_t that indicate the state of the system at time t. The latent representation and state labels are learned simultaneously by predicting the state labels at time $t + \tau$ given the molecular features at time t.

To learn the latent representation \mathbf{z} , SPIB maximizes the loss function

$$\mathcal{L}_{\text{SPIB}} = \mathbf{E}_{\mathbf{X}_{0} \sim \mu, \mathbf{z} \sim p_{\theta}(\mathbf{z} | \tilde{\mathbf{h}}(\mathbf{X}_{0}))} \left[\ln q_{\theta}(s_{\tau} | \mathbf{z}) - \beta \ln \frac{p_{\theta}(\mathbf{z} | \tilde{\mathbf{h}}(\mathbf{X}_{0}))}{r_{\theta}(\mathbf{z})} \right]. \quad (9)$$

The encoder generates the latent representation \mathbf{z} from the input with probability $p_{\theta}(\mathbf{z}|\tilde{\mathbf{h}}(\mathbf{X})) = \mathcal{N}(\mathbf{z}; \mu_{\theta}(\tilde{\mathbf{h}}(\mathbf{X})), \Sigma_{\theta}(\tilde{\mathbf{h}}(\mathbf{X})))$, which is a multivariate normal distribution with learned mean $\mu_{\theta}(\tilde{\mathbf{h}}(\mathbf{X}))$ and learned covariance $\Sigma_{\theta}(\tilde{\mathbf{h}}(\mathbf{X}))$. The decoder takes the latent representation \mathbf{z} and returns the probability $q_{\theta}(s|\mathbf{z})$ of each state label $s; q_{\theta}(s|\mathbf{z})$ is represented by a neural network with output dimension equal to the number of possible state labels. The quantity $r_{\theta}(\mathbf{z})$ is a prior. The state labels are updated during training as follows:

$$s_{\tau} = \operatorname{argmax}_{s} q_{\theta}(s|\mu_{\theta}(\mathbf{h}(\mathbf{X}_{\tau}))).$$
(10)

It is important to note that (10) allows the number of distinct states that are populated to fluctuate (unpopulated states are ignored).

We follow Wang and Tiwary 45 and use a variational mixture of posteriors for the prior⁶¹:

$$r_{\theta}(\mathbf{z}) = \frac{\sum_{i} \omega_{i} p_{\theta}(\mathbf{z}|u_{i})}{\sum_{i} \omega_{i}}, \qquad (11)$$

where ω and u are learned parameters. In this work, we prepared the initial state labels by performing k-means clustering on the CVs learned from VAMPnets based on distances between C_{α} atoms with k = 100 clusters. Training reduces the number of distinct states that are populated to the estimated number of metastable states. Further training details are given in Table S2.

IV. SYSTEMS STUDIED

We examine the performance of geom2vec for analyzing data from long molecular dynamics simulations of three well-characterized fast-folding proteins (chignolin, trp-cage, and villin)⁶². The data for each system is a single, unbiased simulation, which we assume approximately samples the equilibrium distribution. In this section, we introduce each system and briefly describe its structure and dynamics.

A. Chignolin

Chignolin is a 10-residue fast-folding protein with sequence YYDPETGTWY. The folded state consists of three β -hairpin structures that are distinguished by hydrogen bonding between the threeonine side chains and their dihedral angles, which interconvert on the nanosecond timescale⁶³. The trajectory that we analyze is 106 μ s long at 340 K and saved every 0.2 ns⁶². For VAMPnet fitting, we choose $d_o = 3$.

B. Trp-cage

Trp-cage is a 20-residue fast-folding protein⁶⁴; here we study the K8A mutant with sequence DAYAQWLADG-GPSSGRPPPS. Its secondary structure consists of an α helix (residues 2–9), a short 3₁₀-helix (residues 11–14), and a polyproline II helix (residues 17–19); the protein takes its name from Trp6, which is in the core of the folded state. The trajectory is 208 μ s long at 290 K and saved every 0.2 ns⁶². Previous studies of this trajectory generally identified the folded and unfolded states, with varying numbers of intermediates and misfolded states^{46,65}. For VAMPnet fitting, we choose $d_o = 4$.

C. Villin

The 35-residue villin headpiece subdomain $(HP35)^{66}$ is a fast-folding protein with sequence LSDEDFKA-VFGMTRSAFANLPLWnLQQHLnLKEKGLF where nL refers to the unnatural amino acid norleucine. The K65nL/N68H/K70nL mutant was engineered to fold more rapidly⁶⁷. The secondary structure of villin consists of three α -helices at residues 3–10, 14–19, and 22– 32. Villin has a hydrophobic core centered on residues Phe6, Phe10, and Phe17. The trajectory that we study is 125 μ s long at 360 K and saved every 0.2 ns⁶². Previous studies typically identified three states: a folded state, an unfolded state, and a misfolded state¹². Wang et al.⁶⁸ proposed two primary folding pathways, where either the C-terminus or the N-terminus folds first, ultimately leading to the native state. Additionally, a cooperative hydrophobic interaction may facilitate a third folding pathway. For VAMPnet fitting, we choose $d_o = 3$.

D. Training-validation split

Although previous studies used a random split^{8,33,50}. we observe that, due to the strong correlation between successive structures sampled by molecular dynamics simulations, a random split allows networks to achieve high validation scores even when they have memorized the training data rather than learned useful abstractions from it; the models then perform poorly on independent data. Consequently, for all our numerical experiments, we split the data into training and validation sets by time. That is, we select the first 50% of the long trajectory for training and the remainder for validation. If we had access to multiple independent trajectories, randomly choosing trajectories for training and validation would also be appropriate. Some studies split the trajectory into equal segments and draw random segments for training and validation^{46,65} (*k*-fold cross-validation). When there are only two segments, this approach is identical to ours. When every structure is its own segment, one recovers the random split. Intermediate numbers of segments result in intermediate amounts of correlation between the training and validation sets. In cases where we vary the amount of training data, we first select the first 50% of the trajectory and then divide only this half of the trajectory into segments that we draw randomly for training; the second half of trajectory is used as holdout validation set. This approach is fundamentally different from cross-validation and minimizes the correlation between the training and validation datasets.

V. RESULTS

A. VAMPNets

For each system and token mixer architecture pair that we consider, we independently train three VAMPnets using different random number generator seeds (and the training-validation split described in Section IV D). We report the training and validation VAMP-2 scores for the different token mixer architectures for each of the three systems in Figures S1 and S2. For chignolin, GNNs with pooling (summing), SubMixer, and SubFormer reach approximately the same maximum validation score. GNNs with SubMixer and SubFormer require fewer epochs to reach the convergence criteria, but they require more computational time per epoch, as we discuss in Section VI. For both villin and trp-cage, the token mixers generally outperform pooling.

Figure 2 displays the VAMP-2 scores for VAMPNets trained with different training dataset sizes, varying from 5% to 50% of the available data. For chignolin, the GNNs clearly outperform a multilayer perceptron (MLP) that takes distances between pairs of C_{α} atoms as inputs;

there is not a significant difference between pooling and the mixers considered. For trp-cage and villin, the GNN with pooling consistently achieves the lowest scores. The distance-based MLP and GNN with SubMixer perform comparably, presumably because the distances between pairs of C_{α} atoms are sufficient to describe the folding (in contrast to chignolin, as we discuss below). For villin, we combined SubMixer and SubFormer with GVP and augmented them with a global token: this enables the GNNs to outperform the distance-based MLP. The improvement is particularly striking for SubFormer. The models with GVP are more expressive because they use the equivariant features at the token mixing stage and directly mix them with global features after messagepassing. We note that the VAMP scores that we achieve are lower than published ones owing to our choice of the training-validation split (Section IVD) and output dimension d_{o} . While the amount of data does not significantly impact the performance for chignolin, the trp-cage and villin results suggest that additional data would permit achieving higher scores.

To visualize the results, we build histograms of learned CV-value pairs, which we convert to potentials of mean force (PMFs) by taking the negative logarithm (Figures 3, S3, S4, and S5). We also show average values of CVs as functions of physically-motivated coordinates (Figures 4, S6, and S7).

The advantages of the GNN architecture are well illustrated by the results for chignolin. A previous machinelearning study of chignolin identified two slow CVs, one for the folding-unfolding transition and one distinguishing competing folded states⁶³. Our VAMPnets appear to recover these two CVs (Figures 3 and S3; results shown are with SubMixer), distinguishing folded and unfolded states with CV 1 and four folded states with CV 2. To understand the physical differences between the folded states, we plot CVs 1 and 2 as functions of the fraction of native contacts and the χ_1 side chain dihedral of Thr6 or Thr8 (Figure 4). The fraction of native contacts clearly correlates with CV 1, consistent with earlier studies. CV 2 distinguishes the folded states by the configurations of Thr6 and Thr8 side chains, which can each occupy two rotamers, yielding four possible folded states. These side chain dynamics could not be detected by VAMPnets that take backbone internal coordinates as inputs, as is common, or even GNNs limited to backbone atoms (Bonati, Piccini, and Parrinello⁶³ included distances to side chain atoms and then manually curated the inputs). This makes clear the usefulness of the pre-training approach that we take here, which enables treating all the atoms with an architecture that supports both scalar and vector features.



FIG. 2. VAMPnets with various geom2vec architectures. The amount of training data was varied by dividing the training data into 20 trajectory segments of equal length and then randomly selecting the indicated fraction for training. The validation set is held fixed as the second half of each trajectory. Error bars show standard errors over three independent runs.



FIG. 3. Potential of mean force (PMF) of chignolin as a function of the first two CVs learned by a VAMPnet trained with SubMixer. Contours are drawn every 1 kcal/mol. See Figure S3, S4, and S5 for corresponding plots for other architectures and proteins.

B. SPIB

1. Trp-cage

Figure 5 shows a Markov state model based on the states learned for trp-cage with a lag time of 20 ns. There are 13 metastable states. State S_{11} represents the folded ensemble with a well-defined structure. In states S_1 and S_2 the α -helix is folded, and the 3_{10} -helix and polyproline II helix are unfolded. In contrast, in state S_8 the 3_{10} -

helix is folded, and the α -helix and polyproline II helix are unfolded. State S_5 represents a fully unfolded state, which acts as a hub that connects all intermediate states and the folded state, S_{11} . States S_3 , S_4 , S_6 , S_7 , S_9 , S_{10} , and S_{12} are also largely unfolded and differ with regard to the specific conformations of the helices.

We show attention maps of the SubFormer blocks averaged over all structures in Figure 6 and over the structures in each state in Figures S8, S9, and S10. The attention maps are averaged over all heads of the transformer, yielding $(M+1) \times (M+1)$ matrices, where for trp-cage M = 20 represents the number of sequence positions (tokens), and the additional row and column correspond to a global token that encodes pairwise distances between the C_{α} atoms^{57,59}. The learned attention maps can be interpreted in terms of the structure. The residues that are most consistently activated across states and layers are Trp6 (W6) and Asp9–Ser14 (D9–S14), which roughly correspond to the 3_{10} -helix. Tyr3 (Y3), Gln5 (Q5), Gly15 (G15), and Arg16 (R16) are activated in selected states. The attention thus appears to track the packing of residues around Trp6. Across all three layers, the global token remains highly active, consistent with the fact that the metastable states of trp-cage are reasonably well-characterized by the distances between the C_{α} atoms^{14,65}. The fact that tokens other than the global token participate in the attention mechanism again underscores the ability of GNNs to go beyond distances between C_{α} atoms.

2. Villin

Figure 7 shows a Markov state model based on the states learned for villin HP35 with a lag time of 10 ns. There are 11 metastable states. State S_9 represents the



FIG. 4. Chignolin VAMPnet (with SubMixer) CVs as a function of two physical coordinates: the fraction of native contacts and the χ_1 side chain dihedral angle of Thr6 (left) or Thr8 (right). \bar{Q} is the fraction of native contacts smoothed with a 1-ns moving window centered on each time point. We define native contacts as two residues that are three or more positions apart in sequence and have at least one distance between non-hydrogen atoms that is less than 4.5 Å in the crystal structure (5AWL⁶⁹). See Figures S6 and S7 for analogous plots for trp-cage and villin.

fully folded structure, in which all three α -helices are folded and packed compactly. States S_1 and S_3 correspond to fully unfolded states. In states S_0 and S_2 helix 1 is folded, suggesting a pathway in which folding initiates at the N-terminus, while in states S_5 and S_6 helix 3 is folded, suggesting a pathway in which folding initiates at the C-terminus. Both of these pathways are discussed in the literature (see Ref. 68 and references therein).

The attention maps (Figure S11, S12, and S13) exhibit patterns that correspond to features of the folded structure. Notably, the attention consistently focuses on the tokens representing Val9 (V9), Gly11 (G11), Met12 (M12), Arg14 (R14), Pro21 (P21), and Trp23 (W23). These residues correspond roughly to the turns between helices. In the attention maps for states 0, 2, 5, and 6, the tokens corresponding to helix 3 feature prominently; the tokens corresponding to helix 1 are also activated in state 0. The attention maps thus suggest that the network tracks the folding and packing of the helices.

It is interesting to compare our attention maps with those of Ghorbani *et al.*³³ and Huang *et al.*³⁵. Those studies leverage a graph attention network $(GAT)^{70}$ to enhance expressive power and interpretability of their models. GAT computes representations of each node by attending to its one-hop neighboring nodes, which captures local dependencies but fails to model long-range interactions. In contrast, GNN-Transformer hybrids such as SubFormer⁵⁷ allocate short-range interactions to the MP-GNN and use the self-attention mechanism for longrange interactions. This approach not only supports multimodal features (e.g., a global token) but also enables distant nodes to attend to each other, regardless of graph distance. This difference is evident in the attention map patterns: GAT attention maps^{33,35} show predominantly diagonal patterns, reflecting a focus on local neighborhoods, while SubFormer-GVP attention maps reveal vertical, blockwise, and global patterns, reflecting instead a focus on specific amino acids and their long-range interactions.

VI. COMPUTATIONAL REQUIREMENTS

Equivariant geometric GNNs use both invariant and equivariant features to capture the three-dimensional structure of molecules. For typical numbers of features, the memory and time requirements are expensive even for small graphs. To illustrate, we show the memory and time requirements for inference using a TorchMD-ET GNN with a small batch size with varying numbers of hidden channels (numbers of features) for trp-cage (144 non-hydrogen atoms) and villin (272 non-hydrogen atoms) in Figure S14. Even this already requires tens of gigabytes of memory and several seconds; a more complex architecture like ViSNet is expected to increase the memory and time requirements by roughly 50%. The memory and time scale linearly with both the number of atoms in the graph and the batch size. We use a batch size of 5000 for VAMP (except for GVP variants, for which we use 1000) and 1000 for SPIB, making training a GNN on the fly prohibitive, as we discuss in further detail below.

Geom2vec decouples training the GNNs and the networks for the downstream tasks. This allows us to use a small batch size for pretraining the GNNs (which need be done only once), and the networks for the downstream tasks take as inputs the tokens, which are fewer in number than the number of graph nodes. For example, here, the number of graph nodes is the number of non-hydrogen atoms, while the number of tokens is the number of amino acids, which is an order of magnitude smaller.

The computational costs for training VAMPnets with different token mixers are shown in Figure 8. The simplest GNN using pooling is not much more computationally costly than an MLP that takes distances between C_{α} atoms as inputs. The GNNs with token mixers are about an order of magnitude more computationally costly but



FIG. 5. SPIB for trp-cage. All results are obtained from a GNN with SubFormer-GVP token mixer. (top left) PMF as a function of the first two information bottleneck coordinates (IBs). Contours are drawn every 1 kcal/mol. (top right) Same contours colored by SPIB assigned labels. (bottom) Learned Markov State Model. The highlighted structures are chosen randomly from the trajectory. The N-terminus is violet and the C-terminus is red.

still manageable (hundreds of seconds) even without advanced acceleration techniques such as flash-attention or compilation. We expect the memory and computational requirements to scale with token number quadratically for SubFormer and subquadratically for SubMixer (depending on the expansion dimension in the token-mixing blocks); these requirements should scale linearly with respect to embedding dimension and network depth.

To estimate the memory usage and training time for a VAMPnet based on a GNN without pretraining, we consider a TorchMD-ET model with specific settings: a batch size of 1000, a hidden dimension of 64, and 6 layers; we assume 50 epochs are required to converge. For trp-cage, a batch size of 100 required 10.64 GB of memory. Since memory usage and training time should scale linearly with batch size, we estimate that increasing the batch size to 1000 would raise the memory usage to approximately 10.64 GB \times 10 = 106.4 GB. Similarly, we estimate the training time at this larger batch size to be around 1.8 hours (excluding validation). The corresponding numbers for villin, which is about twice the size, are proportionally larger (219.0 GB and 4.0 hours for training). As noted above, employing a more complex architecture like ViSNet would further increase both time and memory requirements by roughly 50%. In the present study, we employ a batch size of 5000 for VAMP (except for GVP variants, as noted above); the batch size for SPIB is 1000, but it generally requires more iterations to converge. While rough, these estimates show that without pretraining equivariant GNNs for analyzing molecular dynamics are beyond the resources available to most researchers. By contrast, with pretraining, they are well within reach (Figure 8).

VII. CONCLUSIONS

In this paper, we use pretrained GNNs to convert molecular conformations into rich vector representations that can then be used for diverse downstream tasks. Decoupling the training of the GNNs and the networks for the downstream tasks dramatically decreases the memory and computational time requirements. Here, we focused on downstream tasks concerned with analyzing dynamics in molecular simulations, specifically VAMP and SPIB. For these tasks, we were able to use equivariant GNNs that take all non-hydrogen atoms of small proteins as inputs for the first time. The results for folding and unfolding of small proteins show that the GNNs use information beyond distances between C_{α} atoms, which are commonly used as input features.

For pretraining, we used a simple denoising task with a dataset of structures of diverse molecules. Given that this dataset was not specific to proteins and/or VAMP and SPIB, we expect the present models to generalize to other classes of molecules and tasks, but quantitative tests on a wider variety of systems and tasks remain to be done. It would be interesting to investigate whether more complex pretraining strategies^{47,71} together with datasets specific to the class of molecules of interest (e.g., those specific to proteins^{48,72,73}) can improve performance. We also explored a variety of token mixers and observed that more complex architectures were able to yield better results, which suggests there is scope for further engineering in this regard; these should be informed by ablation studies.

In our tests, we took care to split the dataset in a way that minimized the correlation between the training and validation datasets, and we believe that this should be standard practice. Because the data consisted of long, unbiased trajectories⁶², there were relatively few events of interest (here, folding and unfolding). Adapting our approach to methods that take short trajectories^{51,74}, which allow for greater control of sampling^{74,75}, is an important area of study for the future.



FIG. 6. SPIB SubFormer-GVP attention. (left) A typical fully folded trp-cage structure (classified as S_{11}) with the central tryptophan residue (Trp6/W6) highlighted. (right three plots) Log-scaled averaged attention maps from three layers of the SubFormer block in the SubFormer-GVP architecture. The sequence is indicated by one-letter amino acid codes, and * represents a global token that encodes pairwise distances between the C_{α} atoms. Tokens from query and key projections are colored according to the row-wise and column-wise sum of layer-wise attention weights. Results shown are for all structures in the trajectory. Results for individual states are in Figures S8, S9, and S10.

ACKNOWLEDGMENTS

We thank D. E. Shaw Research for making the molecular dynamics trajectories available to us. This work was supported by National Institutes of Health award R35 GM136381 and National Science Foundation award DMS-2054306. S.C.G. acknowledges support from the National Science Foundation Graduate Research Fellowship under Grant No. 2140001. Z.P. was supported with funding from the University of Chicago Data Science Institute's AI+Science Research Initiative. This work was completed with computational resources administered by the University of Chicago Research Computing Center, including Beagle-3, a shared GPU cluster for biomolecular sciences supported by the NIH under the High-End Instrumentation (HEI) grant program award 1S10OD028655-0.

DATA AVAILABILITY

Data sharing is not applicable to this article as no new data were created or analyzed in this study. Code for our implementation and examples are available at https: //github.com/dinner-group/geom2vec.

REFERENCES

- ¹B. E. Husic, R. T. McGibbon, M. M. Sultan, and V. S. Pande, "Optimized parameter selection reveals trends in Markov state models for protein folding," Journal of Chemical Physics 145, 194103 (2016).
- ²M. K. Scherer, B. E. Husic, M. Hoffmann, F. Paul, H. Wu, and F. Noé, "Variational selection of features for molecular kinetics," Journal of Chemical Physics **150**, 194108 (2019).
- ³D. Nagel, S. Sartore, and G. Stock, "Selecting features for Markov modeling: A case study on HP35," Journal of Physical Chemistry Letters 14, 6956–6967 (2023).

- ⁴R. E. Arbon, Y. Zhu, and A. S. J. S. Mey, "Markov state models: To optimize or not to optimize," Journal of Chemical Theory and Computation **20**, 977–988 (2024).
- ⁵G. A. Tribello, M. Ceriotti, and M. Parrinello, "Using sketch-map coordinates to analyze and bias molecular dynamics simulations," Proceedings of the National Academy of Sciences **109**, 5196–5201 (2012).
- ⁶M. A. Rohrdanz, W. Zheng, M. Maggioni, and C. Clementi, "Determination of reaction coordinates via locally scaled diffusion map," Journal of Chemical Physics **134**, 124116 (2011).
- ⁷L. Boninsegna, G. Gobbo, F. Noé, and C. Clementi, "Investigating molecular kinetics by variationally optimized diffusion maps," Journal of Chemical Theory and Computation **11**, 5947–5960 (2015).
- ⁸A. Mardt, L. Pasquali, H. Wu, and F. Noé, "VAMPnets for deep learning of molecular kinetics," Nature Communications 9, 5 (2018).
- ⁹F. Noé and F. Nüske, "A variational approach to modeling slow processes in stochastic dynamical systems," Multiscale Modeling & Simulation 11, 635–655 (2013).
- ¹⁰W. Chen, H. Sidky, and A. L. Ferguson, "Nonlinear discovery of slow molecular modes using state-free reversible VAMPnets," Journal of Chemical Physics **150**, 214114 (2019).
- ¹¹H. Wu and F. Noé, "Variational approach for learning Markov processes from time series data," Journal of Nonlinear Science **30**, 23–66 (2020).
- ¹²C. Lorpaiboon, E. H. Thiede, R. J. Webber, J. Weare, and A. R. Dinner, "Integrated variational approach to conformational dynamics: A robust strategy for identifying eigenfunctions of dynamical operators," Journal of Physical Chemistry B **124**, 9354–9364 (2020).
- ¹³Z. Trstanova, B. Leimkuhler, and T. Lelièvre, "Local and global perspectives on diffusion maps in the analysis of molecular systems," Proceedings of the Royal Society A **476**, 20190036 (2020).
- ¹⁴J. Strahan, A. Antoszewski, C. Lorpaiboon, B. P. Vani, J. Weare, and A. R. Dinner, "Long-time-scale predictions from shorttrajectory data: A benchmark analysis of the trp-cage miniprotein," Journal of Chemical Theory and Computation **17**, 2948– 2963 (2021).
- ¹⁵E. H. Thiede, D. Giannakis, A. R. Dinner, and J. Weare, "Galerkin approximation of dynamical quantities using trajectory data," Journal of Chemical Physics **150**, 244111 (2019).
- ¹⁶P. G. Bolhuis, C. Dellago, and D. Chandler, "Reaction coordinates of biomolecular isomerization," Proceedings of the National Academy of Sciences **97**, 5877–5882 (2000).
- ¹⁷A. Ma and A. R. Dinner, "Automatic method for identifying reaction coordinates in complex systems," Journal of Physical



FIG. 7. SPIB for villin. All results are obtained from a GNN with SubFormer-GVP token mixer. (top left) PMF as a function of the first two information bottleneck coordinates (IBs). Contours are drawn every 1 kcal/mol. (top right) Same contours colored by SPIB assigned labels. (bottom) Learned Markov State Model. The highlighted structures are chosen randomly from the trajectory. The N-terminus is violet and the C-terminus is red.

Chemistry B 109, 6769–6779 (2005).

- ¹⁸S. C. Guo, R. Shen, B. Roux, and A. R. Dinner, "Dynamics of activation in the voltage-sensing domain of *Ciona intestinalis* phosphatase Ci-VSP," Nature Communications **15**, 1408 (2024).
- ¹⁹N. S. Herringer, S. Dasetty, D. Gandhi, J. Lee, and A. L. Ferguson, "Permutationally invariant networks for enhanced sampling (PINES): Discovery of multimolecular and solvent-inclusive collective variables," Journal of Chemical Theory and Computation **20**, 178–198 (2023).
- ²⁰N. Thomas, T. Smidt, S. Kearnes, L. Yang, L. Li, K. Kohlhoff, and P. Riley, "Tensor field networks: Rotation-and translationequivariant neural networks for 3D point clouds," arXiv preprint arXiv:1802.08219 (2018).
- ²¹B. Anderson, T.-S. Hy, and R. Kondor, "Cormorant: Covariant molecular neural networks," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems* (Curran Associates Inc., Red Hook, NY, USA, 2019) pp. 14537– 14546.
- ²²S. Batzner, A. Musaelian, L. Sun, M. Geiger, J. P. Mailoa, M. Kornbluth, N. Molinari, T. E. Smidt, and B. Kozinsky, "E(3)equivariant graph neural networks for data-efficient and accurate interatomic potentials," Nature Communications 13, 2453 (2022).
- ²³J. Han, J. Cen, L. Wu, Z. Li, X. Kong, R. Jiao, Z. Yu, T. Xu, F. Wu, Z. Wang, *et al.*, "A survey of geometric graph neural net-

works: Data structures, models and applications," arXiv preprint arXiv:2403.00485 (2024).

- ²⁴J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *International Conference on Machine Learning* (PMLR, 2017) pp. 1263–1272.
- ²⁵P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu, "Relational inductive biases, deep learning, and graph networks," arXiv preprint arXiv:1806.01261 (2018).
- ²⁶B. E. Husic, N. E. Charron, D. Lemm, J. Wang, A. Pérez, M. Majewski, A. Krämer, Y. Chen, S. Olsson, G. de Fabritiis, *et al.*, "Coarse graining molecular dynamics with graph neural networks," Journal of Chemical Physics **153**, 194101 (2020).
- ²⁷A. R. Jamasb, A. Morehead, Z. Zhang, C. K. Joshi, K. Didi, S. V. Mathis, C. Harris, J. Tang, J. Cheng, P. Lio, and T. L. Blundell, "Evaluating representation learning on the protein structure universe," in *The Twelfth International Conference on Learning Representations* (2023).
- ²⁸P. Hermosilla and T. Ropinski, "Contrastive representation learning for 3D protein structures," arXiv preprint arXiv:2205.15675 (2022).
- ²⁹L. Wang, H. Liu, Y. Liu, J. Kurtin, and S. Ji, "Learning hierarchical protein representations via complete 3D graph networks," arXiv preprint arXiv:2207.12600 (2023).
- ³⁰Z. Zhang, M. Xu, A. Jamasb, V. Chenthamarakshan, A. Lozano, P. Das, and J. Tang, "Protein representation learning by geometric structure pretraining," arXiv preprint arXiv:2203.06125 (2023).
- ³¹T. Xie, A. France-Lanord, Y. Wang, Y. Shao-Horn, and J. C. Grossman, "Graph dynamical networks for unsupervised learning of atomic scale dynamics in materials," Nature Communications **10**, 2667 (2019).
- ³²S. Soltani, C. W. Sinclair, and J. Rottler, "Exploring glassy dynamics with Markov state models from graph dynamical neural networks," Physical Review E **106**, 025308 (2022).
- ³³M. Ghorbani, S. Prasad, J. B. Klauda, and B. R. Brooks, "Graph-VAMPNet, using graph neural networks and variational approach to Markov processes for dynamical modeling of biomolecules," Journal of Chemical Physics **156**, 184103 (2022).
- ³⁴B. Liu, M. Xue, Y. Qiu, K. A. Konovalov, M. S. O'Connor, and X. Huang, "GraphVAMPnets for uncovering slow collective variables of self-assembly dynamics," Journal of Chemical Physics **159**, 094901 (2023).
- ³⁵Y. Huang, H. Zhang, Z. Lin, Y. Wei, and W. Xi, "RevGraph-VAMP: A protein molecular simulation analysis model combining graph convolutional neural networks and physical constraints," Methods **229**, 163–174 (2024).
- ³⁶A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17 (Curran Associates Inc., Red Hook, NY, USA, 2017) pp. 6000–6010.
- ³⁷A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," arXiv preprint arXiv:2010.11929 (2020).
- ³⁸T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the 26th International Conference* on Neural Information Processing Systems - Volume 2, NIPS'13 (Curran Associates Inc., Red Hook, NY, USA, 2013) pp. 3111– 3119.
- ³⁹Y. Guo, J. Wu, H. Ma, and J. Huang, "Self-supervised pretraining for protein embeddings using tertiary structures," in *Proceedings of the AAAI Conference on Artificial Intelligence*,



FIG. 8. Computational time for training a single VAMPNet. We employ early stopping and stop training when the training VAMP score does not increase for 1000 batches or the validation VAMP score does not increase for 10 batches. Times reported are averages over three training runs, with validation performed at each step using the second half of each trajectory, as depicted in Figure S2. All times are for training on a single NVIDIA A40 GPU.

Vol. 36 (2022) pp. 6801–6809.

- ⁴⁰S. Zaidi, M. Schaarschmidt, J. Martens, H. Kim, Y. W. Teh, A. Sanchez-Gonzalez, P. Battaglia, R. Pascanu, and J. Godwin, "Pre-training via denoising for molecular property prediction," arXiv preprint arXiv:2206.00133 (2022).
- ⁴¹C. Chen, J. Zhou, F. Wang, X. Liu, and D. Dou, "Structure-aware protein self-supervised learning," Bioinformatics **39**, btad189 (2023).
- ⁴²J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *Proceedings of the 32nd International Conference* on International Conference on Machine Learning - Volume 37, ICML'15 (JMLR.org, Lille, France, 2015) pp. 2256–2265.
- ⁴³ J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in Advances in Neural Information Processing Systems, Vol. 33 (Curran Associates, Inc., 2020) pp. 6840–6851.
- ⁴⁴A. S. Christensen, S. K. Sirumalla, Z. Qiao, M. B. O'Connor, D. G. Smith, F. Ding, P. J. Bygrave, A. Anandkumar, M. Welborn, F. R. Manby, *et al.*, "OrbNet Denali: A machine learning potential for biological and organic chemistry with semiempirical cost and DFT accuracy," Journal of Chemical Physics **155**, 204103 (2021).
- ⁴⁵D. Wang and P. Tiwary, "State predictive information bottleneck," Journal of Chemical Physics **154**, 134111 (2021).
- ⁴⁶D. Wang, Y. Qiu, E. R. Beyerle, X. Huang, and P. Tiwary, "Information bottleneck approach for Markov model construction," Journal of Chemical Theory and Computation **20**, 5352–5367 (2024).
- ⁴⁷Y. Ni, S. Feng, X. Hong, Y. Sun, W.-Y. Ma, Z.-M. Ma, Q. Ye, and Y. Lan, "Pre-training with fractional denoising to enhance molecular property prediction," Nature Machine Intelligence 6, 1169–1178 (2024).
- ⁴⁸Y. Vander Meersche, G. Cretin, A. Gheeraert, J.-C. Gelly, and T. Galochkina, "ATLAS: protein flexibility description from atomistic molecular dynamics simulations," Nucleic Acids Research **52**, D384–D392 (2024).
- ⁴⁹C. X. Hernández, H. K. Wayment-Steele, M. M. Sultan, B. E. Husic, and V. S. Pande, "Variational encoding of complex dynamics," Physical Review E **97**, 062412 (2018).
- ⁵⁰H. Chen, B. Roux, and C. Chipot, "Discovering reaction pathways, slow variables, and committor probabilities with machine learning," Journal of Chemical Theory and Computation 19,

4414-4426 (2023).

- ⁵¹J. Strahan, S. C. Guo, C. Lorpaiboon, A. R. Dinner, and J. Weare, "Inexact iterative numerical linear algebra for neural network-based spectral estimation and rare-event prediction," Journal of Chemical Physics **159**, 014110 (2023).
- ⁵²H. Jung, R. Covino, A. Arjun, C. Leitold, C. Dellago, P. G. Bolhuis, and G. Hummer, "Machine-guided path sampling to discover mechanisms of molecular self-organization," Nature Computational Science **3**, 334–345 (2023).
- ⁵³Y. Wang, T. Wang, S. Li, X. He, M. Li, Z. Wang, N. Zheng, B. Shao, and T.-Y. Liu, "Enhancing geometric representations for molecules with equivariant vector-scalar interactive message passing," Nature Communications **15**, 313 (2024).
- ⁵⁴P. Thölke and G. De Fabritiis, "TorchMD-NET: Equivariant transformers for neural network based molecular potentials," arXiv preprint arXiv:2202.02541 (2022).
- ⁵⁵Z. Pengmei, Z. Shen, Z. Wang, M. Collins, and H. Rangwala, "Pushing the limits of all-atom geometric graph neural networks: Pre-training, scaling and zero-shot transfer," arXiv preprint arXiv:2410.21683 (2024).
- ⁵⁶N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I* (Springer-Verlag, Berlin, Heidelberg, 2020) pp. 213–229.
- ⁵⁷Z. Pengmei, Z. Li, C. chan Tien, R. Kondor, and A. R. Dinner, "Transformers are efficient hierarchical chemical graph learners," arXiv preprint arXiv:2310.01704 (2023).
- ⁵⁸I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit, M. Lucic, and A. Dosovitskiy, "MLP-Mixer: An all-MLP architecture for vision," in *Advances in Neural Information Processing Systems*, Vol. 34 (Curran Associates, Inc., 2021) pp. 24261– 24272.
- ⁵⁹Z. Pengmei and Z. Li, "Technical report: The graph spectral token – enhancing graph transformers with spectral information," arXiv preprint arXiv:2404.05604 (2024).
- ⁶⁰B. Jing, S. Eismann, P. Suriana, R. J. L. Townshend, and R. Dror, "Learning from protein structure with geometric vector perceptrons," in *International Conference on Learning Representations* (2020).

- ⁶¹J. Tomczak and M. Welling, "VAE with a VampPrior," in Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics (PMLR, 2018) pp. 1214–1223.
- ⁶²K. Lindorff-Larsen, S. Piana, R. O. Dror, and D. E. Shaw, "How fast-folding proteins fold," Science **334**, 517–520 (2011).
- ⁶³L. Bonati, G. Piccini, and M. Parrinello, "Deep learning the slow modes for rare events sampling," Proceedings of the National Academy of Sciences **118**, e2113533118 (2021).
- ⁶⁴B. Barua, J. C. Lin, V. D. Williams, P. Kummler, J. W. Neidigh, and N. H. Andersen, "The Trp-cage: optimizing the stability of a globular miniprotein," Protein Engineering, Design and Selection **21**, 171–185 (2008).
- ⁶⁵H. Sidky, W. Chen, and A. L. Ferguson, "High-resolution Markov state models for the dynamics of trp-cage miniprotein constructed over slow folding modes identified by state-free reversible VAMPnets," Journal of Physical Chemistry B **123**, 7999–8009 (2019).
- ⁶⁶J. C. McKnight, D. S. Doering, P. T. Matsudaira, and P. S. Kim, "A thermostable 35-residue subdomain within villin headpiece," Journal of Molecular Biology **260**, 126–134 (1996).
- ⁶⁷J. Kubelka, T. K. Chiu, D. R. Davies, W. A. Eaton, and J. Hofrichter, "Sub-microsecond protein folding," Journal of Molecular Biology **359**, 546–553 (2006).
- ⁶⁸E. Wang, P. Tao, J. Wang, and Y. Xiao, "A novel folding pathway of the villin headpiece subdomain HP35," Physical Chemistry Chemical Physics **21**, 18219–18226 (2019).
- ⁶⁹S. Honda, T. Akiba, Y. S. Kato, Y. Sawada, M. Sekijima, M. Ishimura, A. Ooishi, H. Watanabe, T. Odahara, and K. Harata, "Crystal structure of a ten-amino acid protein," Journal of the American Chemical Society **130**, 15327–15331 (2008).
- ⁷⁰P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," arXiv preprint arXiv:1710.10903 (2017).
- ⁷¹Y.-L. Liao, T. Smidt, M. Shuaibi, and A. Das, "Generalizing denoising to non-equilibrium structures improves equivariant force fields," arXiv preprint arXiv:2403.09549 (2024).
- ⁷²I. Sillitoe, N. Bordin, N. Dawson, V. P. Waman, P. Ashford, H. M. Scholes, C. S. M. Pang, L. Woodridge, C. Rauer, N. Sen, M. Abbasian, S. Le Cornu, S. D. Lam, K. Berka, I. Varekova, R. Svobodova, J. Lees, and C. A. Orengo, "CATH: increased structural coverage of functional space," Nucleic Acids Research 49, D266–D273 (2021).
- ⁷³I. Barrio-Hernandez, J. Yeo, J. Jänes, M. Mirdita, C. L. M. Gilchrist, T. Wein, M. Varadi, S. Velankar, P. Beltrao, and M. Steinegger, "Clustering predicted structures at the scale of the known protein universe," Nature **622**, 637–645 (2023).
- ⁷⁴J. Strahan, J. Finkel, A. R. Dinner, and J. Weare, "Predicting rare events using neural networks and short-trajectory data," Journal of computational physics **488**, 112152 (2023).
- ⁷⁵J. Strahan, C. Lorpaiboon, J. Weare, and A. R. Dinner, "BAD-NEUS: Rapidly converging trajectory stratification," Journal of Chemical Physics **161**, 084109 (2024).
- ⁷⁶K. T. Schütt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko, and K.-R. Müller, "SchNet-a deep learning architecture for molecules and materials," Journal of Chemical Physics 148, 241722 (2018).
- ⁷⁷J. Gasteiger, J. Groß, and S. Günnemann, "Directional message passing for molecular graphs," arXiv preprint arXiv:2003.03123 (2020).
- ⁷⁸S. Villar, D. W. Hogg, K. Storey-Fisher, W. Yao, and B. Blum-Smith, "Scalars are universal: Equivariant machine learning, structured like classical physics," Advances in Neural Information Processing Systems **34**, 28848–28863 (2021).
- ⁷⁹M. Weiler, M. Geiger, M. Welling, W. Boomsma, and T. S. Cohen, "3D steerable CNNs: Learning rotationally equivariant features in volumetric data," in *Advances in Neural Information Processing Systems*, Vol. 31 (Curran Associates, Inc., 2018).
- ⁸⁰K. T. Schütt, O. T. Unke, and M. Gastegger, "Equivariant message passing for the prediction of tensorial properties and molecular spectra," arXiv preprint arXiv:2102.03150 (2021).

⁸¹K. Everett, L. Xiao, M. Wortsman, A. A. Alemi, R. Novak, P. J. Liu, I. Gur, J. Sohl-Dickstein, L. P. Kaelbling, J. Lee, and J. Pennington, "Scaling exponents across parameterizations and optimizers," arXiv preprint arXiv:2407.05872 (2024).

Appendix A: GNN architectures

1. Equivariant GNNs

Assuming $\mathbf{R} = (\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_N) \in \mathbb{R}^{3N}$ defines the molecular conformation, where $\mathbf{r}_i \in \mathbb{R}^3$ represents the threedimensional (3D) coordinates of the *i*-th atom, two of the most important symmetries for a GNN to obey are translation and rigid-body rotation invariances. These invariances require that the output of the GNN, $f(\mathbf{R})$, remains unchanged when the molecule is translated by a vector $\mathbf{t} \in \mathbb{R}^3$, i.e., $f(\mathbf{R}) = f(\mathbf{R} + \mathbf{t})$, or when the molecule undergoes a rigid body rotation represented by a rotation matrix $P \in SO(3)$, i.e., $f(\mathbf{R}) = f(P\mathbf{R})$. Translation invariance can be easily realized by using the relative displacements or internal coordinates of atoms. On the other hand, rotational invariance and equivariance can be realized in two ways:

- 1. representing the relative atom positions by internal coordinates (e.g., bond distances, bond angles, and dihedral angles) as in conventional classical force fields ("scalar-based network")^{53,54,76,77};
- 2. encoding the relative displacement vectors with spherical harmonics, which are then propagated with tensor products ("group-equivariant network")²⁰⁻²².

The two approaches are theoretically and practically equivalent $5^{3,54,78}$, but group-equivariant networks need to store the features for each irreducible representation and are more computationally costly owing to the tensor products. Therefore, we chose to employ scalar-based GNNs.

2. TorchMD-ET architecture

The TorchMD-ET architecture 54 is structured around three components that process and encode molecular geometric information effectively.

1. Embedding layer: encodes atomic types and interatomic distances using exponential radial basis functions (eRBFs). Each distance d_{ij} within a cutoff r_{cut} is transformed by

$$eRBF_k(d_{ij}) = \phi(d_{ij}) \exp\left(-\beta_k(\exp(-d_{ij}) - \mu_k)^2\right),\tag{A1}$$

where $\phi(d_{ij})$ is a cosine cutoff function ensuring smooth transitions to zero beyond $r_{\rm cut}$.

2. Modified attention mechanism: incorporates edge information through an extended dot-product attention mechanism that integrates interatomic distances with distance kernels DK derived from eRBFs (\odot denotes element-wise multiplication):

$$A = \text{SiLU}\left(\sum_{k=1}^{F} Q_k \odot K_k \odot DK_k\right) \cdot \phi(d_{ij}).$$
(A2)

The attention mechanism weights are then used to compute scalar features and filters $q_{\alpha i}, s_{\alpha i j} \in \mathbb{R}^F$ ($\alpha = 1, 2, 3$) for the update layer:

$$s_{1ij}, s_{2ij}, s_{3ij} = V_j \odot DV \tag{A3}$$

$$q_{1i}, q_{2i}, q_{3i} = W\left(\sum_{j \in \mathcal{N}(i) \setminus i} A_{ij} \cdot s_{3ij}\right) \tag{A4}$$

where V_i and DV are attention value and distance projections analogous to Q, K, and DK.

3. Update layer: updates both scalar (Δx_i) and vector (Δv_i) features using $q_{\alpha i}$ and $s_{\alpha i j}$:

$$\Delta x_i = q_{1i} + q_{2i} \odot (U_1 \mathbf{v}_i \cdot U_2 \mathbf{v}_i) \tag{A5}$$

$$\Delta \mathbf{v}_i = q_{3i} \cdot U_3 \mathbf{v}_i + \sum_j \left(s_{1ij} \odot \mathbf{v}_j + s_{2ij} \odot \frac{\mathbf{r}_i - \mathbf{r}_j}{\|\mathbf{r}_i - \mathbf{r}_j\|} \right).$$
(A6)

This combines scalar and directional vector information describing the local atomic environment.

3. ViSNet

ViSNet⁵³ builds on TorchMD-ET by incorporating additional scalar features into its architecture. These additional features include information about angles, dihedral angles, and improper dihedral angles. This information is computed in an efficient fashion by first associating with each atom information about its neighbors:

$$\mathbf{u}_{ij} = \mathbf{r}_{ij} / \|\mathbf{r}_{ij}\|,\tag{A7}$$

$$\mathbf{v}_i = \sum_{j \in \mathcal{N}(i) \setminus i} \mathbf{u}_{ij},\tag{A8}$$

$$\mathbf{w}_{ij} = \mathbf{v}_i - (\mathbf{v}_i \cdot \mathbf{u}_{ij})\mathbf{u}_{ij},\tag{A9}$$

where \mathbf{u}_{ij} is the unit vector pointing from atom i to neighboring atom j, and \mathbf{v}_i is the sum of all such unit vectors around atom i. Then the inner product

$$\mathbf{w}_{ij} \cdot \mathbf{w}_{ji} = \sum_{m \in \mathcal{N}(i) \setminus i} \sum_{n \in \mathcal{N}(j) \setminus j} \cos(\varphi_{mijn})$$
(A10)

represents the dihedral angle (φ_{mijn}) information around inner atoms i and j, with neighbors indexed by m and n, respectively.

Gated equivariant block 4.

The output layer of the networks consists of gated equivariant blocks^{79,80}, each of which combines the scalar (x_i) and vector (\mathbf{v}_i) features from graph node *i* of the previous layer (Algorithm 1).

Algorithm 1 Gated equivariant block (GEB)	
Require: $x_i \in \mathbb{R}^{d_i}, \mathbf{v}_i \in \mathbb{R}^{3d_i}$	
1: $v_i^1, \mathbf{v}_i^2 \leftarrow \ W\mathbf{v}_i\ , W\mathbf{v}_i$	$\triangleright v_i^1 \in \mathbb{R}^{d_i}, \mathbf{v}_i^2 \in \mathbb{R}^{3d_o}$
2: $x_i \leftarrow \operatorname{Concat}(x_i, v_i^1)$	$ hinspace{-1.5}{ ho} x_i \in \mathbb{R}^{2d_i}$
3: $x_i \leftarrow W(\text{SiLU}((Wx_i + b))) + b$	$ hinspace{} hinspace{} x_i \in \mathbb{R}^{2d_o}$
4: $[x_i, g_i] \leftarrow x_i$	\triangleright split x_i into $x_i, g_i \in \mathbb{R}^{d_o}$
5: $x_i \leftarrow \text{SiLU}(x_i)$	
6: $\mathbf{v}_i \leftarrow g_i \odot \mathbf{v}_i^2$	▷ scales vector features, maintains equivariance
7: return x_i , \mathbf{v}_i	$\triangleright x_i \in \mathbb{R}^{d_o}, \mathbf{v}_i \in \mathbb{R}^{3d_o}$

Appendix B: Geometric vector perceptron

Geometric vector perceptron (GVP) was one of the first architectures to incorporate vector features for protein modeling⁶⁰. Here, we use the GVP architecture as an equivariant token mixer to combine scalar and vector features (Algorithm 2).

Algorithm 2 Geometric vector perceptron (GVP)

Require: $x_i \in \mathbb{R}^{d_i}, \mathbf{v}_i \in \mathbb{R}^{3d_i}$	
1: $\mathbf{v}_h \leftarrow W_h \mathbf{v}_i$	\triangleright project vector features, $\mathbf{v}_i \in \mathbb{R}^{3d_h}$
2: $x_h \leftarrow \ \mathbf{v}_h\ _2$	\triangleright compute the (row-wise) norm of projected vectors, $x_h \in \mathbb{R}^{d_h}$
3: $x_{h+d_i} \leftarrow \text{Concat}(x_h, x_i)$	$\triangleright \ x_{h+d_i} \in \mathbb{R}^{d_h+d_i}$
4: $x_m \leftarrow W_m x_{h+d_i} + b_m$	$ hinspace{1.5} ho x_m \in \mathbb{R}^{d_o}$
5: $x'_i \leftarrow \sigma(x_m)$	$ hinspace x_i' \in \mathbb{R}^{d_o}$
6: $\mathbf{v}_o \leftarrow W_o \mathbf{v}_h$	\triangleright project vector features to \mathbb{R}^{3d_o}
7: $\mathbf{v}'_i \leftarrow \sigma(\ \mathbf{v}_o\ _2) \odot \mathbf{v}_o$	\triangleright column-wise multiplication with vector gate (on row-wise norm), $\mathbf{v}_i' \in \mathbb{R}^{3d_o}$
8: return x'_i, \mathbf{v}'_i	

Algorithm 3 geom2vec					
Require: Atomic numbers z_i , positions $\mathbf{r}_i \in \mathbb{R}^3$ $(i = 1,, N)$, coarse-grained mapping $\mathcal{S} = \{S_1,, S_M\}$, mixer					
method {None, SubFormer, SubMixer, SubGVP}, global features $\mathbf{G} \in \mathbb{R}^{d_g}$					
1: $x_i^{\text{MP}}, \mathbf{v}_i^{\text{MP}} \leftarrow \text{GNN}(z_i, \mathbf{r}_i)$	\triangleright atomistic featurization, $x_i^{\text{MP}} \in \mathbb{R}^{d_k}, \mathbf{v}_i^{\text{MP}} \in \mathbb{R}^{3d_k}$				
2: $x_m^{\text{MP}}, \mathbf{v}_m^{\text{MP}} \leftarrow \sum_{i \in S_m} x_i^{\text{MP}}, \sum_{i \in S_m} \mathbf{v}_i^{\text{MP}}$	\triangleright coarse-graining with mapping $\mathcal{S}, x_m^{\text{MP}} \in \mathbb{R}^{d_k}, \mathbf{v}_m^{\text{MP}} \in \mathbb{R}^{3d_k}$				
3: if None then					
4: $x^{\text{MP}}, \mathbf{v}^{\text{MP}} \leftarrow \sum_m x_m^{\text{MP}}, \sum_m \mathbf{v}_m^{\text{MP}}$	\triangleright sum over indices <i>m</i> for direct pooling, $x^{MP} \in \mathbb{R}^{d_k}, \mathbf{v}^{MP} \in \mathbb{R}^{3d_k}$				
5: $x, \mathbf{v} \leftarrow \text{GEB}(x^{\text{MP}}, \mathbf{v}^{\text{MP}})$	\triangleright Algorithm 1				
6: else if SubFormer then					
7: $x_m, \mathbf{v}_m \leftarrow \text{GEB}(x_m^{\text{MP}}, \mathbf{v}_m^{\text{MP}})$	\triangleright apply GEB to input features				
8: if global token then					
9: $g \leftarrow \mathrm{MLP}(\mathbf{G})$	\triangleright project global features G to \mathbb{R}^{d_k}				
10: $x \leftarrow \text{TransformerEncoder}([x_m, g])$	\triangleright pass regular and global tokens through the transformer encoder				
11: else					
12: $x \leftarrow \text{TransformerEncoder}(x_m)$	\triangleright pass only the regular tokens through the transformer encoder				
13: end if					
14: else if SubMixer then					
15: $x_m, \mathbf{v}_m \leftarrow \text{GEB}(x_m^{\text{MP}}, \mathbf{v}_m^{\text{MP}})$	\triangleright apply GEB to input features				
16: if global token then					
17: $g \leftarrow \mathrm{MLP}(\mathbf{G})$	\triangleright project global features G to \mathbb{R}^{d_k}				
18: $x \leftarrow \text{MLPMixer}([x_m, g])$	\triangleright pass regular and global tokens through the MLP-Mixer				
19: else					
20: $x \leftarrow \text{MLPMixer}(x_m)$	\triangleright pass only the regular tokens through the MLP-Mixer				
21: end if					
22: else if SubGVP then					
23: $x_m^{\text{GVP}}, \mathbf{v}_m^{\text{GVP}} \leftarrow \text{GVP}(x_m^{\text{MP}}, \mathbf{v}_m^{\text{MP}})$	\triangleright Algorithm 2				
24: if subformer or submixer then					
25: $x \leftarrow \text{subformer or submixer}(x_m^{\text{GVP}})$) \triangleright apply subformer or submixer block to GVP features				
26: end if					
27: end if					
28: $x \leftarrow \mathrm{MLP}(x)$					
29: return <i>x</i>	$\triangleright x \in \mathbb{R}^{d_o}$				

Appendix D: Training details and hyperparameters

For pre-training, we use the Orbnet Denali dataset from Ref. 44, which consists of 2.3 million conformations of small to medium size molecules. These conformations are obtained with semi-empirical methods and contain nonequilibrium geometries, alternative tautomers, and non-bonded interactions. For pretraining, we randomly take 10,000 configurations as the validation set and the remainder as the training set.

TABLE S1. Hyperparameters for pretrained networks		TABLE S2. Hyperparameters for SPIB		
Hyperparameter	TorchMD-ET and ViSNet	Hyperparameter	Shared	
d	64, 128, 256, 384	Batch size	1000	
# of MP layers	6	Learning rate	0.0002	
# of attention heads	8	Optimizer	AdamAtan2	
Batch size	100	Training patience	5	
Epochs	10	MLP hidden dimension	64	
# of RBFs	64	MLP activation function	SiLU	
$r_{\rm cut}$ (Å)	5, 7.5	Embedding dimension (d)	64	
Learning rate	0.0005	Dropout	0.2	
Optimizer	AdamW (AMSGrad)	Batch normalization	False	
Noise level (Å)	0.2	Label refinement frequency	5	
	·	Architectures	SubFori	mer/SubMixer
		GVP		Yes
		# of GVP layers		3
		# of transformer/mixer layers		3
		Expansion factor		2
		Global token		True
			Villin	Trp-cage
		Trajectory stride	2	4
		Training lag time (ns)	20	10

TABLE S1. Hyperparameters for pretrained networks

TABLE S3. Hyperparameters for VAMPnets. For GVP variants tested on villin, the architecture is consistent with the ones used in SPIB tasks.

Hyperparameter	Chignolin	Villin	Trp-cage
Batch size	5000	5000/1000	5000/1000
Learning rate	0.0002	0.0002	0.0002
Optimizer	$AdamAtan2^{81}$	AdamAtan2	AdamAtan2
Maximum epochs	20	20	20
Training patience	5	500	500
Validation patience	2	10	10
Validation interval	5	50	50
MLP hidden dimension	256	128/64	128/64
MLP activation function	SiLU	SiLU	SiLU
Trajectory stride	1	2	4
Training lag time (ns)	4	20	10
Embedding dimension (d)	256	128/64	128/64
Dropout	0.2	0.2	0.2
Batch normalization	False	False	False
Output dimension (d_o)	2	3	4
Architectures	MLP/SubFormer/SubMixer	MLP/SubFormer/SubMixer	MLP/SubFormer/SubMixer
GVP	No	Yes	No
Expansion factor	2	2	2
Global token	False	True	False

Appendix E: Supplementary Figures



FIG. S1. Training curves for VAMPnets. Each curve represents the mean of three training runs. Shaded regions indicate standard error over three runs.



FIG. S2. Validation curves for VAMPnets. Each curve represents the mean of three training runs. Shaded regions indicate standard error over three runs.



FIG. S3. PMFs as a function of VAMP CVs for chignolin. From top to bottom, VAMPnets were trained with no token mixer (Sum), SubMixer, or SubFormer. Each column shows the result from a single training run. Contours are drawn every 1 kcal/mol.



FIG. S4. PMFs as a function of VAMP CVs for trp-cage. From top to bottom, VAMPnets were trained with no token mixer (Sum), SubMixer, or SubFormer. Each column shows the result from a single training run. Contours are drawn every 1 kcal/mol.



FIG. S5. PMFs as a function of VAMP CVs for villin. From top to bottom, VAMPnets were trained with no token mixer (Sum), SubMixer, SubMixer-GVP, SubFormer, or SubFormer-GVP. Each column shows the result from a single training run. Contours are drawn every 1 kcal/mol.



FIG. S6. Trp-cage VAMPnet (with SubMixer) CVs as a function of two physical coordinates: C_{α} RMSD of helix 1 (residues 2–9) and C_{α} RMSD of helix 2 (residues 11–14). The C_{α} RMSDs were computed with respect to the PDB structure 2JOF⁶⁴.



FIG. S7. Villin VAMPnet (with SubFormer) CVs as a function of two physical coordinates: C_{α} RMSD of helices 1 and 2 (residues 3–10 and 14–19), and C_{α} RMSD of helices 2 and 3 (residues 14–19 and 22–32). The C_{α} RMSDs were computed with respect to the PDB structure $2F4K^{67}$.



FIG. S8. Log-scaled attention weight heatmaps for trp-cage SPIB states 0 to 4 from three layers of SubFormer-GVP. Each subplot displays attention weights with color-coded tick labels based on normalized sums. Colorbars indicate log-scaled attention values.



FIG. S9. Log-scaled attention weight heatmaps for trp-cage SPIB states 5 to 9 from three layers of SubFormer-GVP. Each subplot displays attention weights with color-coded tick labels based on normalized sums. Colorbars indicate log-scaled attention values.



FIG. S10. Log-scaled attention weight heatmaps for trp-cage SPIB states 10 to 12 from three layers of SubFormer-GVP. Each subplot displays attention weights with color-coded tick labels based on normalized sums. Colorbars indicate log-scaled attention values.



FIG. S11. Log-scaled attention weight heatmaps for villin SPIB states 0 to 3 from three layers of SubFormer-GVP. Each subplot displays attention weights with color-coded tick labels based on normalized sums. Colorbars indicate log-scaled attention values.



FIG. S12. Log-scaled attention weight heatmaps for villin SPIB states 4 to 7 from three layers of SubFormer-GVP. Each subplot displays attention weights with color-coded tick labels based on normalized sums. Colorbars indicate log-scaled attention values.



FIG. S13. Log-scaled attention weight heatmaps for villin SPIB states 8 to 10 from three layers of SubFormer-GVP. Each subplot displays attention weights with color-coded tick labels based on normalized sums. Colorbars indicate log-scaled attention values.



FIG. S14. Comparison of time and GPU memory usage of TorchMD-ET in training and inference modes on the trp-cage (left) and villin (right) subsets, each consisting of 2000 frames with 100 frames per batch. The measurements are done without specific objective functions for usage benchmarking purposes. The model's performance is evaluated across varying numbers of hidden channels (64, 128, 256, 384) and layers (1–6). The heatmaps on the left show GPU memory allocated in gigabytes (GB), while the heatmaps on the right depict time usage in seconds. The time represents one forward pass for inference mode and one forward plus one backward pass for training mode. Measurements were performed on an NVIDIA A100 GPU with 40G of memory. Missing values are due to out-of-memory errors for certain configurations.