Online Boosted Gaussian Learners for in-situ Detection and Characterization of Protein Folding States in Molecular Dynamics Simulations

Harshita Sahni*, Hector Carrillo-Cabada†, Ekaterina Kots‡, Silvina Caino-Lores§, Jack Marquez§, Ewa Deelman¶, Michel Cuendet‖, Harel Weinstein‡, Michela Taufer§, Trilce Estrada*

Affiliation: *Universty of New Mexico, †Intuit, ‡Weill Cornell Medical Center, §University of Tennessee, ¶University of Southern California, ‖Swiss Institute of Bioinformatics

2023 IEEE 19th International Conference on e-Science (e-Science) | 979-8-3503-2223-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/E-SCIENCE58273.2023.10254895

***Abstract*—Molecular Dynamics (MD) simulations are a crucial tool for understanding how proteins fold. In its easiest form, MD simulations can be scaled through data parallelism, this means that multiple folding trajectories can be spawned and executed in parallel, facilitating a more efficient exploration of the protein folding space. However, due to data dependencies, the analysis of MD simulations remains largely as a centralized process. In this work, we propose a data parallel, lightweight technique to learn the characteristics of protein folding states in MD simulations. Contrary to other methods, ours can differentiate relevant states in a single protein folding trajectory without requiring centralized global knowledge of the protein dynamics. As its processing and memory overheads are negligible (in the order of milliseconds per window of frames, and kilo bytes respectively) this technique can be coupled with the simulation for in-situ analysis.**

I. Introduction

Proteins are complex molecules composed of amino acid residues [1] that perform critical functions in cells, like providing structure, catalyzing metabolic reactions, and transporting molecules. Advances in X-ray crystallography and even in Machine Learning [2] have allowed us to extract proteins' tertiary structure, which provides insight on how proteins perform their function. However, this structure is not static, it folds and unfolds to perform the protein's functions. Changes in factors like pH, temperature, and composition of a solution can make a protein go through a number of molecular events that may induce conformational changes. Structural rearrangements, binding events, and protein associations [3] are examples of such changes. Understanding why and how conformation changes occur is crucial, because they provide information about the biological, self-assembly functions, and molecular mechanisms of the protein [4].

Molecular Dynamics (MD), which is based on statistical physics, is used to simulate the folding process of proteins over a period of time [5]. Each time step, which is termed as a **frame**, defines the conformational state of the protein at a specific point in time. The chain or collection of frames forms a **trajectory**. MD simulations have been successfully scaled up in HPC environments, as they are task and data-parallel. That means, they can be decomposed into multiple independent trajectories with their own data, and processed in parallel. Proteins and other biomolecules have incredibly intricate atomistic dynamics that span time periods from subpicosecond to hours [6], [7]. Depending on its sampling rate and length, each simulation can produce gigabytes of data [8]. Analysis of MD simulations includes finding specific molecular events and the conformation changes that a protein undergoes. Traditional analyses rely on building a global view of all the conformations in all of the trajectories for a specific molecular system [9]–[11]. This form of analysis represents a scalability bottleneck since individual simulations tend to explore only a relatively small number of the possible folding conformations of a protein [12]. Thus, multiple trajectories are needed before even being able to start analyzing a system.

Coupling simulations with data analytics on the same node (i.e., in-situ analysis) has shown to be effective in multiple domains [13]–[17]. However, the requirements to make this approach work are not trivial: to begin with, sharing resources between simulation and analysis necessarily constrains the amount of computation, memory, and I/O that are available for the analysis. Large overheads would negatively impact the simulation's performance, defeating the whole purpose of in-situ analysis. Second, communication among analysis processes is prohibitive. Since the expectation is to be able to scale the analysis at the same rate as the simulations, the analysis must also be perfectly parallel. This point constrains the analysis to derive global insights from local information only.

Contributions. In this paper we present our method for in-situ analysis of MD simulations through Boosted Gaussian Learners. We designed this method with the *in-situ* constraints in mind: it works as a stream, its memory and CPU overheads are negligible, it does not require communication with other analysis processes, and it is able to detect and model protein folding states with only local information.

The paper is organized as follows: in section 2 we discuss the state of the practice with respect to MD analysis; in section 3 we describe our approach to learn the characteristics of

protein folding states in MD simulations; in section 4 we evaluate our results in terms of accuracy and performance; section 5 summarizes our work and future research.

II. State of the Practice

In recent years, machine learning has been increasingly applied to the analysis of protein simulations, and in this section, we provide a review of the most common approaches to studying molecular events and detecting relevant folding states in protein simulations.

Dimensionality reduction techniques. There are methods that use dimensionality reduction and rely on finding specialized features in order to capture the slow movements of the protein and study molecular events [18]. For example, work has been done [19]–[22] to represent protein trajectories using diffusion maps, which makes it simpler to analyze and define numerous states in the protein. These maps can also be used to identify significant patterns that capture the intrinsic structure of the system. Furthermore, methods like t-SNE [23], Sketch Maps [24], [25], UMAP [26] and Isomaps [27] have been used to project the data onto a lower dimensional representation for understanding the behavior of the system and identifying molecular changes. Time-lagged independent component analysis (tICA) [28] is the gold standard dimensionality reduction technique used in representing otherwise highly complex multidimensional conformational spaces of protein dynamics. tICA is used for encapsulating the slowest-moving components and ignoring the fast or irrelevant features [9]–[11]. Principal Component Analysis (PCA) is also used to detect important motions in the protein system [29], [30]. However, tICA is more suitable for the task, since PCA selects the features with high variance and tICA identifies the slow reaction coordinates or finds a maximally slow subspace at a given lag time, which is useful when the goal is to detect molecular events from MD data. Plenty of work [31]–[34] has used tICA to detect and track how the protein moves through several conformational states. However, tICA can be misleading in the detection of relevant molecular changes where faster degrees of freedom are useful in representing the changes. Another disadvantage is that the selection of lag time is important to properly discretize the space [18]. The main drawback of using dimensionality reduction techniques is that to find accurate analyses of conformational states, it is crucial to mix various evaluation criteria rather than relying solely on one of them. Also, it is hard to select the correct latent space representation that separates the states of the protein [35], [36] and significant manual exploration is needed to achieve satisfactory results.

Clustering. Another alternative to identify metastable states of the protein system is through clustering. Work by Keller et al. [37] puts together different kinetic and geometric clustering techniques to study the various states in a protein trajectory. Sittel et al. [38] developed a density-based clustering technique to study the conformational states with a complexity of NlogN (where N is the number of input points/frames). Of particular importance in this domain is Markov State Model (MSM) [10], [11], [39], which is used to examine conformational dynamics of a system for predicting different behaviors in protein trajectories [40]. Several other works in the literature also used tICA and MSM for state analysis of protein folding [41]–[43]. In MSM, the molecule's conformation defines the states, and the probability that the molecule will change from one conformation to another defines transitions between the states. It is one of the most widespread strategies used to cluster the distinct states for short trajectories [44], [45]. The main drawback of these clustering techniques is that one needs to maintain caution when selecting the hyperparameters. For example, predetermining the number of clusters, choosing collective variables, and selecting the right algorithm for a specific molecule. Substantial manual intervention is necessary for these algorithms to converge to accurate folding states.

Other Machine Learning approaches targeted classifying protein trajectories into specific folding states. Zhou, Dong, and Tao [46] employed Decision Trees (DT) and Artificial Neural Networks (ANN) for classifying ligand unbound and bound states from MD trajectories of the PDZ2 protein. Hayatshahi et al. [47] worked with Deep Neural Networks (DNN) and DT for classifying different states of the PDZ3 protein. Approaches such as Time lagged Autoencoder (TAE), Encoder Map, and Variational Dynamic Encoder (VDE) are used for finding collective variables or features to represent the high dimensional data [48]. They exploit the reconstruction losses in order to find useful embeddings to study the molecular events. In addition, Oliver et al. [49] used various traditional ML approaches, namely principal component analysis (PCA), random forest (RF), autoencoders, restricted Boltzmann machines, and multilayer perceptron, to analyze various conformational states within the soluble-protein calmodulin. All these techniques are centralized and contrary to our method, they require all trajectories to be computed before they can be used for analysis.

In situ analysis of MD simulations. There are some frameworks that offer ways by which analysis of MD simulations can be performed in situ. However, these techniques require major efforts to screen the conformational states through the protein structures which are generated at run-time. Also, the state analysis is still relegated to sequential processing, when the simulation is completed. For example, VMD [50] can be used for in-situ analysis [51], but it only provides visualization insights rather than automated analysis. A more general approach, proposed by Tu et al. [52] developed a map-reduce-based framework called the HiMach, which develops a platform to analyze MD simulations in situ. In this work, they performed All-to-All RMSD (i.e., computing the average distance between atoms of superimposed proteins compared to a reference structure) within a sliding window. This approach has two drawbacks. (1) All-to-All RMSD is computationally expensive and (2) evaluating All-to-All RMSD between a sliding window might not provide an accurate analysis for the complete set of structures within the trajectory. Dordaneh et al. [53] monitor secondary structures at runtime to analyze MD data.

III. Method

When a protein transitions into different conformational states, the process is not as clean and definitive as we would like. On the contrary, the protein may partially move from one conformation to another, and then return to the previous fold, or transition completely into a different conformational state. Given a long enough simulation, the protein can move around the free energy landscape and transition into multiple states back and forth. In order to successfully determine the different conformational states adopted by the protein through the simulation, the detection mechanism needs to be able to adapt on the fly. That is, it needs to be able to accurately identify changes in the distribution as new data is collected. On top of that, to be able to analyze MD simulations in-situ, we require an analysis solution with minimum compute and memory overhead, and no other communication than collecting the input data. Our MD in-situ analysis, based on online boosted Gaussian learners, meets all of those requirements.

In this section, we describe the two main components of our method: Section III-A explains the steps we take to project the original data into a much lower dimension. Then, Section III-B presents our online analysis pipeline for which we incrementally generate a graph that characterizes different conformational states that are observed as the simulation progresses.

A. Extraction of Spatio-Temporal Features from Local Protein Dynamics

The main idea of our method is to model the behavior of the protein dynamics when we assume the protein fold is in a specific state. That is, to determine what is the *normal* range of movement of this conformation. To do so, we do the following: 1) process every frame to extract its collective variables, 2) collect a window of frames to capture the spatio-temporal correlation of residues of interest, and 3) perform Non-negative Matrix factorization over the window of frames.

Input. Our analysis process receives from the simulation one topology file and trajectory information. The topology is in the Protein Data Bank (PDB) format, which contains a variety of annotations, including atomic connectivity and 3D atomic coordinates. This file is used only once at the beginning of the analysis to determine the atoms to be used during the rest of the simulation. The trajectory data can be in either XTC, which is the Gromacs compressed trajectory format, or DCD, which is a cross-platform binary trajectory format used by CHARMM, NAMD, LAMMPS, and OpenMM. The trajectory data contains updated atomic 3D positions used to calculate the specific protein conformation at a given frame. To ingest the input data, we use mdtraj, an open library for the manipulation of configurational data from MD simulations [54].

Extraction of collective variables. Collective variables (CVs), also referred to as reaction coordinates are features used to describe each frame of the MD simulation. Instances of CVs are inter-atomic distances, dihedral angles, the radius of gyration of atoms in the system, among others. CVs and the specific residues of interest are determined by scientists with the intention of capturing a specific molecular process; for example, the opening or closing of a pocket, the transport of atoms across a membrane, or the folding of a particular structure. In this work, we use all-pairs of inter-atomic distances, a.k.a an Euclidean distance matrix [55], in Angstroms, among all pairs of alpha-Carbon atoms of the residues of interest. Even though other distance metrics are available, the Euclidean distances are robust to rotational changes in the protein and hence serve as the best collective variable for detecting molecular events. Figure 1 shows three frames of the BBL simulation: the protein conformations and their respective distance matrices for residues of interest. Each row and column correspond to a particular alpha-Carbon, and the intensity of the cell is the distance between each pair of atoms.

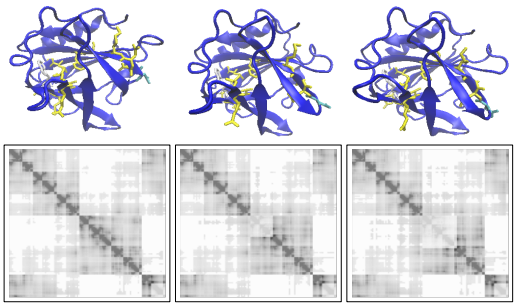


Fig. 1: Input: atomic distance between residues of interest

Frame collection. In order to capture temporal information from the protein dynamics, we collect a consecutive set of frames, that we call a window. Empirically, we determined that 50 frames per window gave us the best trade-off between efficiency and preservation of information. A window of 50 frames is also a small enough number given that our typical simulations range from 6,000 to 60,000 frames. More than 50 frames can be used if simulations are performed at a very fine-grained temporal resolution. However, less than 50 would result in suboptimal performance for our learning algorithm, as discussed in Section III-B. Finally, to process a window of frames, we flatten the distance matrix into a vector and concatenate the frames to produce a matrix X of the size number of residues squared times 50.

Non-negative Matrix Factorization (NMF) [56] is a matrix decomposition technique that factorizes an input matrix X into two smaller matrices: a feature matrix W, and a coefficient matrix H. W and H exhibit clustering properties and can be used to identify underlying patterns or structures in the data. The key constraint in NMF is that all values must be larger than or equal to zero. This constraint is easily satisfied in our setup, since atoms can only have positive Euclidean distances. Formally, NMF uses the Non-negative Double Singular Value Decomposition [57] to minimize the Frobenius norm between X and WH:

$$d_{Fro}(X, WH) = \frac{1}{2}||X - WH||^2_{Fro} \quad (1)$$

Our input, the matrix X, contains m rows and n columns, where m is the square of the number of residues of interest

(i.e., spatial information), and n is the number of frames (i.e., temporal information). Given a hyperparameter $k < min(m, n)$, W is a matrix of size $(m \times k)$, that can be thought of as k protein conformation archetypes comprising relevance of distances among residues when the protein is in this particular state. The higher the value of a distance pair, the more important the relationship between these residues is to characterize the particular protein conformation. H is a $(k \times n)$ coefficient matrix that defines the importance of the k components to each frame. The k components of W and H are obtained in an additive way. Components are superimposed and never subtracted, resulting in a series of sparse interpretable models.

To select the latent size k for the NMF decomposition, we test values ranging from the square root of the minimum dimension in X and the number of frames in X (i.e., 50). We stop the search as soon as the maximum squared residual is less than 0.002. That is, when the maximum reconstruction error between $(X - WH)$ is less than 0.05 Angstroms.

B. Online Boosted Gaussian Learners

For the in-situ scenario where we are positioning this work on, there are hard constraints in terms of resource utilization, as well as in the type and amount of information that is available during the analysis, especially when we are investigating a new protein or protein mutant. We are required to detect relevant conformation changes at run time, and produce global insights with only local information. That is, we need to be able to identify relevant folding states in a multidimensional space so that the analysis is able to discover only frame by frame.

To account for these important characteristics of our problem we took loose inspiration from Online Boosting algorithms [58]. In regular Boosting [59], multiple base models, also known as weak learners, are built in an incremental way, where a new model is trained from instances weighted by the errors of the last model. Instances that were misclassified by the last model account for half the weight of the training instances for the new model. Through this process, new models become more specialized than their predecessors. Then, a voting mechanism is used to determine the final prediction. A disadvantage of Boosting for our scenario, is that all data needs to be available at once (e.g., all simulations from all trajectories in a system). On the other hand, Online algorithms process every data instance (e.g a single simulation frame) as it is seen by the algorithm for the first and only time. In Online Boosting, as new instances arrive, they are classified in order by the trained models, when one of them produces a misclassification, the instance receives a higher weight to be used to update the next model.

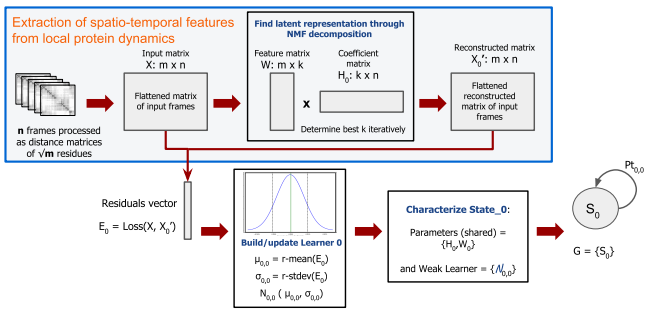
Differently from other Boosting algorithms where the base models are usually decision trees or decision stumps, we choose to use Gaussian models as the base of our weak learners. This decision was made because we observed that when a window of frames contains mostly *similar* folding conformations, and we use NMF to project and reconstruct the window of frames, its residuals are Normally distributed. We also observed that when the protein is transitioning between different folding states, these residuals are either heavily skewed or show a marked distribution change. With these two observations, we hypothesize that we could fit a Gaussian distribution to the residuals, and use it to identify when the protein was changing conformations. We define our Gaussian Learners as parameterized decision functions that determine whether a sample of reconstruction errors $e = e_1, e2, ...e_n$ can be *explained* by its mean, variance, and maximum reconstruction error:

$$GL = f(e, N(\mu, \sigma^2), M_k, k, \alpha_\mu, \alpha_s) : \\ True \text{ if } \begin{cases} \mu + z\frac{\sigma}{\sqrt{k}} \leq \overline{e} \leq \mu + z\frac{\sigma}{\sqrt{k}} \\ \alpha_s \geq \frac{M_k/(k-1)}{\sum (e_i - \overline{e})^2/(n-1)} \\ max(x) < 1\mathring{A} \end{cases} \tag{2}$$

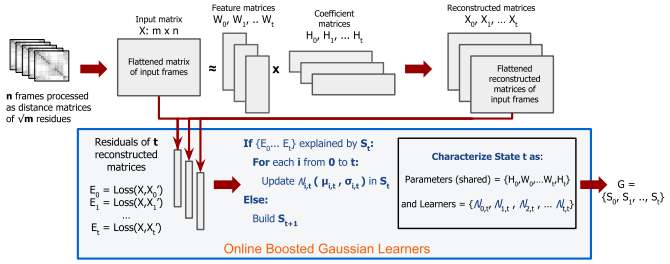
where $z = zscore(1 - \alpha_\mu/2)$. Since our method is designed to work online and does not store data from other windows, we use Welford's online algorithm for the calculation of the variance in a single pass [60] to calculate parameters μ, σ^2 and M_k (see equation 3), where μ is the accumulated mean of k instances, σ^2 is the biased variance estimator, s^2 is the unbiased variance estimator, and M_k is the accumulated squared sum of mean differences. In this way, for every Gaussian Learner, we only need to store four accumulation variables and its parameters, making it extremely cheap to create an ensemble of learners.

$$\begin{aligned} \delta_1 &= e_k - \mu_{k-1} \\ \mu_k &= \mu_{k-1} + \delta_1/k \\ \delta_2 &= e_k - \mu_k \\ M_k &= M_{k-1} + \delta_1\delta_2 \\ \sigma_k^2 &= M_k/k \\ s_k^2 &= M_k/(k-1) \end{aligned} \tag{3}$$

The main idea of our Online Boosted Gaussian Learners (OBGL) algorithm is this: we separate the learning stages into States, and contrary to other Boosting methods, we have the ability to generate additional features to be used by new models. We define a State as a collection of weak Gaussian learners (GL) and projection parameters obtained from NMF (W, H). Then, we incrementally build States based on the Normal distribution of our training window: as new data arrives, if it can be explained by any of the trained States, the window is assigned to that State. If it does not, then the data window is marked as an uncertainty region and is used to update the last State. If the number of consecutive uncertainty windows is larger than a threshold, then a new State is created. We keep a transition matrix to aid in the selection of the most likely State when more than one State can explain the window. We can also perform a probabilistic State assignment and produce a hierarchy of states. The process continues for as long as new data is collected. Figure 2 depicts the State creation in two phases: for State 0 and for an arbitrary State t. In the next paragraphs, we explain in more detail how this sequence works.



(a) Building the initial state (S_0)



(b) Generalizing to t states through boosted Gaussian learners

Fig. 2: Method overview for one state (a) and for multiple states (b)

Our input is a window of frames, for each frame we extract its Euclidean distance matrix and flatten it into a vector that becomes a row in the window matrix X_0, as described in Section III-A. We initialize a graph of states G and use the first window of frames to build State (S_0), as depicted in Figure 2a. The process is as follows:

- We perform a NMF decomposition of X_0 into parameters W_0 and H_0 and produce a reconstructed $X_0' = W_0 \cdot H_0$. We calculate a reconstruction loss vector $E_0 = (X_0 - X_0')^2$
- We use the vector of residuals E_0 to build our first Gaussian Learner $GL_{0,0}$ where we model the probability of a frame, as expressed by its residual e_k, to belong to State S_0 as $P(e_k|S_0) = N(\mu_{0,0}, \sigma^2_{0,0})$.
- State S_0 is characterized by its Gaussian Learner $GL_{0,0} = N(\mu_{0,0}, \sigma^2_{0,0})$

Once State S_0 is built, it is used to determine if subsequent windows of frames can be explained by its single Gaussian Learner. If a window cannot be explained, then it is marked as an uncertainty region. After the number of consecutive uncertainty windows surpasses a threshold τ then a new state is created. Generalizing to t states, the incremental creation of a state S_t for an arbitrary window X, as depicted in Figure 2b, is as follows:

- We perform a new NMF decomposition of X into parameters W_t, H_t.
- We use the stored $W_0, H_0, ...W_{t-1}, H_{t-1}$ and the newly generated W_t, H_t to produce a series of reconstructed $X_0' = W_0 \cdot H_0$ to $X_t' = W_t \cdot H_t$ and their respective reconstruction loss vectors $E_0 = (X - X_0')^2, E_1 = (X - X_1')^2 ..., E_t = (X - X_t')^2$. Note that $W_0, H_0, ...W_{t-1}, H_{t-1}$ are shared with the $t-1$ previous states.
- For each vector E_0 to E_t we build their respective Gaussian Learner for state t $S_t = \{GL_{0,t}, GL_{1,t}, ...GL_{t,t}\}$.
- Finally, we add S_t to our states graph $G = \{S_0, S_1, ...S_t\}$ and add a row and a column to the transition matrix.

Once multiple states have been created, subsequent windows are evaluated in the order given by the probability of each state r times the probability of transitioning from the last explained state q. That is, from $S_q \rightarrow S_r$, as: $P(S_r)P(r|q)$. We can make state assignments in two different ways: 1) hard assignment, where a window is assigned to the first state that is able to explain it, or 2) soft assignment, where a window can be assigned to the top *few* states that are able to explain the window, where "*few*" can range from 2 to 4 for practical purposes. If a window cannot be explained by any of the existing states, then it is marked as an uncertainty region and used to update *only* the last state. We do this, to ensure the stationarity of states that have been successful at explaining other protein conformation changes. After a certain consecutive number of uncertainty regions are detected, a new state is created (we call this a break-point). In order to keep our memory overhead low, we allow for state removal. When we are at a break-point, we assess whether the last state was used to explain any previous window. If the state was not used, then it is removed from our graph, and all of its parameters are erased. In this case, we assume that it was noise that caused the creation of the unused state.

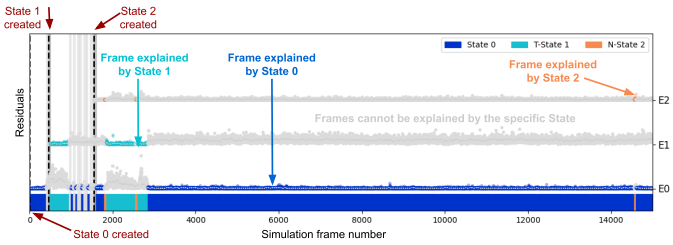


Fig. 3: State assignment per simulation frame

Figure 3 shows a trace, after the simulation is completed, of the temporal events detected in-situ by the analysis. In red, we show the point in time where the different states were created. The bar at the bottom of the image, shows to which state each frame was assigned to. For example, blue corresponds to frames assigned to S_0. In the vertical axis, we show the residuals per frame. For example, E_1 corresponds to the residuals generated by projecting a window X into W_1 and H_1 and computing its reconstruction loss as $E_1 = (X - X_1')^2$. In this figure, it is evident the change in variance in E_0 when the new state S_1 was generated. We can also see when the protein transitioned from $S_0 \rightarrow S_1 \rightarrow S_0 \rightarrow S_1 \rightarrow S_0$. Given the frequency at which the protein remained in one state versus the others, we define S_0 as an actual state, S_1 as a transition state, and S_3 as noise.

IV. Evaluation

To test the accuracy, scalability and robustness of our method we evaluated its performance on three major protein systems: B cell translocation gene (BTG1) and its mutants, Bovine beta-lactoglobulin (BBL), and Opsin comprising a total of 122 trajectories. We quantitatively validated our in-situ results with a tICA clustering produced offline [9], [41], and we measured performance in terms of compute overhead and memory usage.

A. Protein Systems

Protein	Trajs.	Res.	Res. CV	Frames
Globulin BBL	6	162	12	33300
Opsin	43	326	21	316953
B cell translocation gene (BTG1) Mutants				
Wild-Type	23	129	8	100600
E50K	12	129	8	82000
R68L	6	129	8	84300
A8RT	1	129	8	60000
Q36H-E50K	15	129	8	100000
Q36H_D107N	11	129	8	179000
A84E	1	129	8	60500
F40E	1	129	8	60500
Q36L	1	129	8	35000
Q45P	1	129	8	60200
Q36N	1	129	8	35000

TABLE I: Protein systems used for validations. Trajs: number of trajectories per system, Res: total number of residues, Res CV: residues of interests, Frames: Total number of frames in the system

To validate the generality of our method across a variety of folding landscapes, we tested our method on three major protein systems:

Globulin BBL. Members of the Globulin family of proteins are expressed in humans and all other mammals and share an overall spherical shape. The beta-globulins are essential in blood function physiology. We performed MD Simulations of BBL with the OpenMM 7.3 software using the CHARMM36 all-atom force field at 310K and a pH of 8, sampling the trajectories at a rate of 10 picoseconds per frame. We investigated the transition between open and closed conformations of the loop spanned by residues 83-91 to ascertain the role of Glu89 in the conformational transition. The transition is considered complete or a new molecular event is examined when both the side-chain orientation of residues (89 & 90) and their backbone positions are changing [61], [62]. We investigated six trajectories for this system with a total length varying from 2000 to 10000 frames. This system has a total of 1286 atoms and 162 residues.

Opsin belongs to the superfamily of G protein-coupled receptors (GPCR). It binds to light-reactive agents like Retinal to support many physiological processes in living organisms (e.g., light detection in circadian cycles, and phototaxis). The MD Simulation of the Opsin protein system was performed with OpenMM 7.4 at 310K using the CHARMM36m force-field. Each trajectory was sampled at a rate of 80ps per frame [63]. The trajectories encode well-known conformational changes observed in the cytoplasmic ends of the 5th and 6th transmembrane helices [63]. We investigated two variants of Opsin. This system is composed of 326 residues and 5258 atoms (just protein). For the first system, we collected 42 trajectories, each containing 7499 frames. While the second Opsin variant has one trajectory with 1995 frames.

B cell translocation gene (BTG1) system is associated with the group of antiproliferative proteins. The main function of this family is to perform cell cycle regulation and other vital cellular processes. For BTG1 and its mutants, the MD simulations were performed with OpenMM 7.4 using the CHARMM36 force field parameters with 310 K temperature, sampled with a rate of 10 picoseconds per frame. For this system and its mutants, we investigated the protein region formed by helices alpha-2 and alpha-4 [43]. We investigated 82 trajectories obtained for 11 different mutants of BTG1. The length of the trajectories ranges from 1000 to 63910 frames. The system consists of 129 residues and the number of atoms in the simulated systems range from 2061 to 2077.

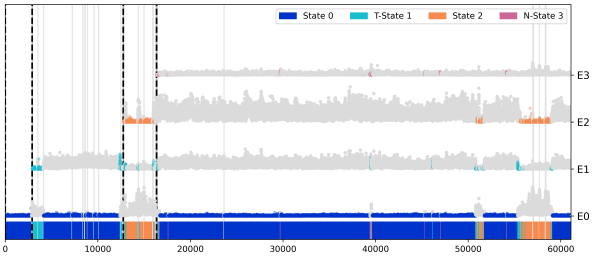
Table 1 provides additional details on the specific proteins and mutants, as well as the specific number of trajectories, number of frames, and number of residues of interest used in the simulations.

B. Accuracy Evaluation

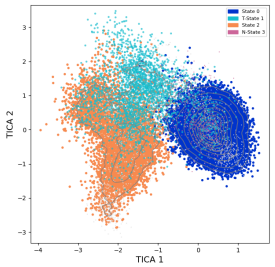
To quantify the accuracy of our method, we validate our results through comparison with an independently generated tICA map described in Section 2, above. The dimensionality reduction method tICA [9] has widespread application in representing otherwise highly complex multidimensional conformational spaces of protein dynamics. To obtain the information needed for quantifying the accuracy of our method, the MD simulation trajectories, including all the spawned trajectories, are projected onto the 2D tICA space (shown in Figures 4 - 5). The tICA space is computed only **after** the simulation for a specific protein system has been completed (i.e., this information is not available while performing in-situ analysis). Projection of the MD simulation data, given enough sampling, on the tIC1 & tIC2 space creates a 2D representation of the free energy surface of the conformational transition in the protein of interest. As such, it provides us with important thermodynamic information. For example, the location and number of the free energy minimums: indicating stable conformations, and maximums: indicating energy barriers that the system needs to overcome to proceed with the conformational transition. Briefly outlined, the offline protocol for clustering protein conformation states is as follows: 1) collect all trajectories for a particular protein system, 2) build a matrix of atomic distances between the residues of interest for all of the trajectories of a particular system, 3) Perform the tICA decomposition and select the first two components, 4) Use the projected data into a 2D space and perform k-means clustering on the 2D-TICA coordinates, 5) Refine the final clustering assignment using an agglomerative clustering based on the transition matrix of the system. With this information,

we were able to quantify the accuracy of our method. To determine if the states found by our method were coherent with the tICA space, we projected each trajectory onto its tICA space. In the next figures, we show examples of our in-situ state identification with respect to the tICA clustering for trajectories of varying degrees of complexity.

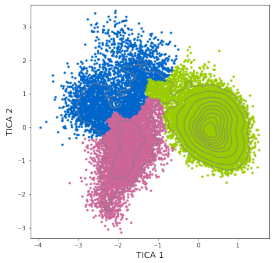
Figures 4 and 5 show two representative trajectories. Part (a) shows the State trace produced at run time. This trace shows the time when new states were created (vertical dotted lines), uncertainty regions (vertical gray bars), states to which a frame was assigned (bottom colored bar), residuals for each state (vertical axis, denoted by E_0 to E_7), and the distinction between regular states and transition (T-State) and noise (N-state). Part (b) shows the state assignment produced by our in-situ method after being projected into the tICA space for the system. It is important to recall that this 2D space is not available a priori to our analysis method. Part (c) shows the clustering of the tICA space produced after all trajectories were collected. We look for clusters that are in the right position and have a consistent shape with respect to tICA.



(a) Trace of in-situ state identification



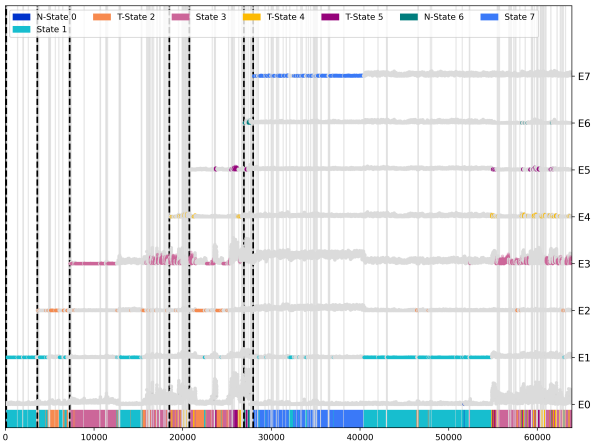
(b) In-situ clustering



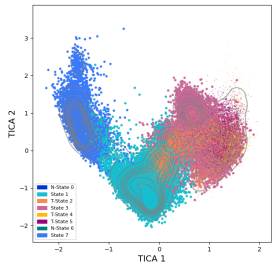
(c) Offline tICA clustering

Fig. 4: A83T - validation of our in-situ state assignment vs. its offline tICA clustering.

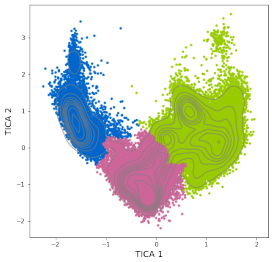
Figure 4 represents the results for a BTG1 mutant. This is a long trajectory, comprising 60,000 frames sampled at 10 ps per frame, for a total of 600 nanoseconds. This simulation exhibits three well-formed clusters in the tICA space. Our method was able to correctly identify State 0 and State 2 as very distinctive components and identified the transition State 1 as something in between 0 and 2. A few conformations with a high probability of transitioning between states were classified as noise. This information is also very relevant since conformations in this category can be used to spawn new simulations and produce potentially richer explorations. The trace also shows the very well-defined transition between $S_0 \rightarrow TS_1 \rightarrow S_0 \rightarrow S_2 \rightarrow S_0 \rightarrow S_2 \rightarrow S_0 \rightarrow S_2 \rightarrow S_0$. This transition pattern shows how our approach preserves the stationarity of learned states, as older states are able to explain frames further away in the trajectory.



(a) Trace of in-situ state identification



(b) In-situ clustering



(c) Offline tICA clustering

Fig. 5: A84E - validation of our in-situ state assignment vs. its offline tICA clustering.

Figure 5 analyzes our single more complex trajectory. This is another BTG1 mutant consisting again of 60,000 frames and 600 nanoseconds. The energy landscape shows irregular areas of high density. The tICA clustering resulted in three major clusters. Our method was able to accurately identify them as State 1, State 3, and State 7. Their location and boundaries match very well with the validation. Our method, however, produced additional transition and noise states. Of particular interest is transition State 2, which shows sections of the trajectory with a higher probability to transition between states 1 and 7. Proving us with insight into which specific conformations are more useful to more efficiently explore the energy landscape.

Table 2 presents a summary of our evaluation. Based on the tICA clustering, as described above, we compared our results and computed the total number of True Positives (TP), defined as those where: a state was found and the size and shape of the cluster were consistent with tICA; False Positives (FP): where a state was overlapping another state, or its size and shape were not consistent with tICA; False Negatives (FN):

where a state was not detected, usually shown as two states merged into one. Note that we cannot compute True Negatives, as some trajectories may comprise only incomplete fragments of the energy landscape. Thus, a trajectory may never visit one or more states (i.e., the protein never adopted a particular conformation within a specific trajectory) at all.

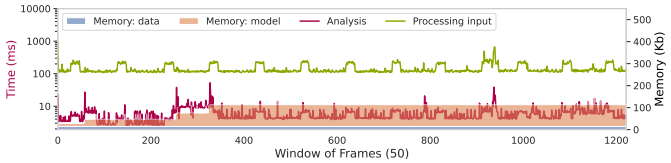
Protein	**TP**	**FP**	**FN**
BBL	10	1	2
Opsin	81	24	1
B cell translocation gene (BTG1) Mutants			
Wild-Type	26	1	2
E50K	15	0	4
R68L	7	0	1
A8RT	3	0	0
Q36H-E50K	28	1	0
Q36H_D107N	12	0	0
A84E	3	0	0
F40C	2	0	0
Q36L	2	1	0
Q45P	3	1	0
Q36N	1	0	0
Total	193	29	10

TABLE II: Protein systems used for validation. TP: True Positives, FP: False Positives, FN: False Negatives

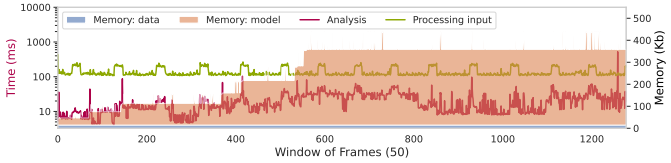
In total, we were able to identify 193 clusters out of 203 and produced 29 false positive detections, 24 of which were concentrated in the Opsin system. We argue that this result is due to the very irregular landscape produced by Opsin and possibly because our hyperparameters were too sensitive for this system. Still, we were able to correctly identify 81 of its states and missed only one. Tuning hyperparameters α_{mu} and α_s can make our method more or less sensitive and produce improved results in terms of false positives and false negatives. In fact, by doing that, we were able to reduce the false positives in E50K and Wild Type to zero. However, we strive to keep all of the hyperparameters fixed across systems and trajectories. This is because for a new system, if we expect to be able to analyze it in-situ from scratch, then we will not have the luxury of hyperparameter tuning. Therefore, all results in Table 2 were performed with the exact same analysis setup.

C. Performance Analysis

In order to run our analysis alongside the simulations, we integrated our method in the in-situ framework in [64]. However, it would be computationally prohibitive to prototype, refine, and finally execute our analysis for the number of long trajectories we had collected. Hence, we ran the analysis as an emulated in-situ environment, where trajectory frames were loaded as if they would come from the simulation code, but without re-running all the simulations over and over again. To measure performance, we used an Intel Xeon 2 Core processor at 2.2GHz and 12GB RAM. We measured performance in terms of RAM and elapsed CPU time. Note that this is a realistic evaluation, as our analysis is designed to run on a single node alongside embarrassingly parallel simulations. This is possible because we completely eliminate the need for communication among analysis processes, and only local information is used to determine the folding states.



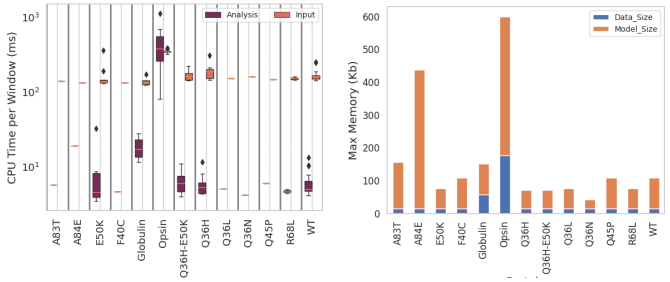
(a) Protein System BTG1 mutant: A83T



(b) Protein system BTG1 mutant: A84E

Fig. 6: Trace of representative protein folding trajectories showing various performance scenarios per window of frames

Figure 6 shows performance traces for the two representative trajectories shown in the analysis: A83T and A84E. The trace distinguishes between memory used for data as a blue area, memory used for our method as an orange area, CPU time for ingesting input data as a green line, and CPU time for analysis as a burgundy line. Since a processing step is done every 50 frames, the figures show performance data per window of frames and not per single frame. The first thing worth noticing is that the memory required for data is kept constant at all times, that is we collect 50 frames, analyze them, and discard them before collecting the next 50. Keeping a bound on the amount of data memory needed was one of our main requirements for making this analysis in-situ. In this way, we guarantee that no matter how long the simulation is, the memory use will remain bounded. The bulk of memory required for our models grows linearly with the number of states that are identified (except for the transition matrix, which is quadratic with respect to the number of states). This memory is not bounded in the same way as the data memory, however, in the worst case the maximum total amount of memory required by our most expensive trajectory did not exceed 600 KB, which is well within the margin of resources that can be spared for analysis in most modern computer systems. Figure 7b shows the breakdown of maximum memory used for data and maximum memory used for models, per protein system. In terms of CPU time, Figure 7 shows the compute spikes every time a new State is identified. The complexity of the analysis for each window is worst-case quadratic on the number of states. For a states graph, G is $O(|G|^2)$; this is because every State has an incremental number of Gaussian learners for a total of $(|G|(|G|+1))/2$ possible evaluations. However, every learner is evaluated at a constant time. Again, the number of states is always going to be relatively small, ranging from one to a couple of dozen. The figure also shows that data ingest is computationally more expensive than the analysis. However, total computation time is, in most cases, within 1 second per window of frames, that is 50 simulation steps can be analyzed within 1 second. In the worst case, for our largest system, the execution time per frame was



(a) CPU time per window (b) Maximum memory used

Fig. 7: CPU and RAM overhead for in-situ analysis

around 10 seconds, which is still a very low overhead for MD simulations. Figure 7a shows a boxplot of execution time split between analysis and input processing. The figure shows the variance per protein system. The figure shows how in most cases (except Opsin) input processing is more expensive than analysis.

V. Conclusion and Future Work

Our method of online boosted Gaussian learners offers a way to monitor molecular events in MD protein simulations at run time. We are able to successfully identify different folding states of the protein through its trajectory without the need for prior knowledge or extensive hyperparameter tuning. Our method is robust to simulations of varying complexity, including variations in sampling rate, number of residues, and simulation length. Finally, but most importantly, we are able to eliminate any and all communication between analysis processes, keeping very low compute and memory overheads, making this approach suitable for in-situ analysis. Since we can use only local information, our analysis can run seamlessly alongside embarrassingly parallel simulations. Future work includes autonomously selecting hyperparameters based on the degrees of freedom of the specific system being analyzed, as well as integrating this analysis in other MD in-situ frameworks.

VI. Acknowledgement

This research was supported by the National Science Foundation (NSF) under grant numbers 1757207, 1741057, 1841758, 2138811 and 2223704; the Oak Ridge Leadership Computing Facility under allocation CSC427; the Extreme Science and Engineering Discovery Environment (XSEDE) under allocation TG-CIS200053; and IBM through a Shared University Research Award

References

[1] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter, "The shape and structure of proteins," in *Molecular Biology of the Cell. 4th edition*. Garland Science, 2002.

[2] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko *et al.*, "Highly accurate protein structure prediction with alphafold," *Nature*, vol. 596, no. 7873, pp. 583–589, 2021.

[3] D. Tobi and I. Bahar, "Structural changes involved in protein binding correlate with intrinsic motions of proteins in the unbound state," *Proceedings of the National Academy of Sciences*, vol. 102, no. 52, pp. 18 908–18 913, 2005.

[4] A. Koide, S. Abbatiello, L. Rothgery, and S. Koide, "Probing protein conformational changes in living cells by using designer binding proteins: application to the estrogen receptor," *Proceedings of the National Academy of Sciences*, vol. 99, no. 3, pp. 1253–1258, 2002.

[5] M. Karplus and J. A. McCammon, "Molecular dynamics simulations of biomolecules," *Nature structural biology*, vol. 9, no. 9, pp. 646–652, 2002.

[6] K. Henzler-Wildman and D. Kern, "Dynamic personalities of proteins," *Nature*, vol. 450, no. 7172, pp. 964–972, 2007.

[7] J. R. Lewandowski, M. E. Halse, M. Blackledge, and L. Emsley, "Direct observation of hierarchical protein dynamics," *Science*, vol. 348, no. 6234, pp. 578–581, 2015.

[8] Y. Wang, J. M. L. Ribeiro, and P. Tiwary, "Machine learning approaches for analyzing and enhancing molecular dynamics simulations," *Current opinion in structural biology*, vol. 61, pp. 139–145, 2020.

[9] Y. Naritomi and S. Fuchigami, "Slow dynamics in protein fluctuations revealed by time-structure based independent component analysis: the case of domain motions," *The Journal of chemical physics*, vol. 134, no. 6, p. 02B617, 2011.

[10] C. R. Schwantes and V. S. Pande, "Improvements in markov state model construction reveal many non-native interactions in the folding of ntl9," *Journal of chemical theory and computation*, vol. 9, no. 4, pp. 2000–2009, 2013.

[11] G. Pérez-Hernández, F. Paul, T. Giorgino, G. De Fabritiis, and F. Noé, "Identification of slow molecular order parameters for markov model construction," *The Journal of chemical physics*, vol. 139, no. 1, p. 07B604_1, 2013.

[12] V. Botu and R. Ramprasad, "Adaptive machine learning framework to accelerate ab initio molecular dynamics," *International Journal of Quantum Chemistry*, vol. 115, no. 16, pp. 1074–1083, 2015.

[13] W. Ma, S. Xu, H. Liu, and Y. Bai, "Mass spectrometry methods for in situ analysis of clinical biomolecules," *Small Methods*, vol. 4, no. 4, p. 1900407, 2020.

[14] L. Wang, N. Limodin, A. El Bartali, J.-F. Witz, R. Seghir, J.-Y. Buffiere, and E. Charkaluk, "Influence of pores on crack initiation in monotonic tensile and cyclic loadings in lost foam casting a319 alloy by using 3d in-situ analysis," *Materials Science and Engineering: A*, vol. 673, pp. 362–372, 2016.

[15] T. Hirono, M. Takahashi, and S. Nakashima, "In situ visualization of fluid flow image within deformed rock by x-ray ct," *Engineering Geology*, vol. 70, no. 1-2, pp. 37–46, 2003.

[16] Y. Guo, W. D. Compton, and S. Chandrasekar, "In situ analysis of flow dynamics and deformation fields in cutting and sliding of metals," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 471, no. 2178, p. 20150194, 2015.

[17] S. Hans, C. Boutin, E. Ibraim, and P. Roussillon, "In situ experiments and seismic analysis of existing buildings. part i: Experimental investigations," *Earthquake Engineering & Structural Dynamics*, vol. 34, no. 12, pp. 1513–1529, 2005.

[18] F. Sittel and G. Stock, "Perspective: Identification of collective variables and metastable states of protein dynamics," *The Journal of chemical physics*, vol. 149, no. 15, p. 150901, 2018.

[19] B. Nadler, S. Lafon, R. R. Coifman, and I. G. Kevrekidis, "Diffusion maps, spectral clustering and reaction coordinates of dynamical systems," *Applied and Computational Harmonic Analysis*, vol. 21, no. 1, pp. 113–127, 2006.

[20] A. L. Ferguson, A. Z. Panagiotopoulos, I. G. Kevrekidis, and P. G. Debenedetti, "Nonlinear dimensionality reduction in molecular simulation: The diffusion map approach," *Chemical Physics Letters*, vol. 509, no. 1-3, pp. 1–11, 2011.

[21] S. B. Kim, C. J. Dsilva, I. G. Kevrekidis, and P. G. Debenedetti, "Systematic characterization of protein folding pathways using diffusion maps: Application to trp-cage miniprotein," *The Journal of chemical physics*, vol. 142, no. 8, p. 02B613_1, 2015.

[22] L. Haghverdi, F. Buettner, and F. J. Theis, "Diffusion maps for high-dimensional single-cell analysis of differentiation data," *Bioinformatics*, vol. 31, no. 18, pp. 2989–2998, 2015.

[23] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[24] G. 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*, vol. 109, no. 14, pp. 5196–5201, 2012.

[25] M. Ceriotti, G. A. Tribello, and M. Parrinello, "Demonstrating the transferability and the descriptive power of sketch-map," *Journal of chemical theory and computation*, vol. 9, no. 3, pp. 1521–1532, 2013.

[26] F. Trozzi, X. Wang, and P. Tao, "Umap as a dimensionality reduction tool for molecular dynamics simulations of biomacromolecules: a comparison study," *The Journal of Physical Chemistry B*, vol. 125, no. 19, pp. 5022–5034, 2021.

[27] V. Spiwok and B. Králová, "Metadynamics in the conformational space nonlinearly dimensionally reduced by isomap," *The Journal of chemical physics*, vol. 135, no. 22, p. 224504, 2011.

[28] B. E. Husic and V. S. Pande, "Markov state models: From an art to a science," *Journal of the American Chemical Society*, vol. 140, no. 7, pp. 2386–2396, 2018.

[29] S. A. M. Stein, A. E. Loccisano, S. M. Firestine, and J. D. Evanseck, "Principal components analysis: a review of its application on molecular dynamics data," *Annual Reports in Computational Chemistry*, vol. 2, pp. 233–261, 2006.

[30] M. Ernst, F. Sittel, and G. Stock, "Contact-and distance-based principal component analysis of protein dynamics," *The Journal of chemical physics*, vol. 143, no. 24, p. 12B640_1, 2015.

[31] S. Schultze and H. Grubmuller, "Time-lagged independent component analysis of random walks and protein dynamics," *Journal of Chemical Theory and Computation*, vol. 17, no. 9, pp. 5766–5776, 2021.

[32] G. Pérez-Hernández and F. Noé, "Hierarchical time-lagged independent component analysis: computing slow modes and reaction coordinates for large molecular systems," *Journal of chemical theory and computation*, vol. 12, no. 12, pp. 6118–6129, 2016.

[33] Y. Naritomi and S. Fuchigami, "Slow dynamics in protein fluctuations revealed by time-structure based independent component analysis: the case of domain motions," *The Journal of chemical physics*, vol. 134, no. 6, p. 02B617, 2011.

[34] V. Spiwok and P. Kříž, "Time-lagged t-distributed stochastic neighbor embedding (t-sne) of molecular simulation trajectories," *Frontiers in Molecular Biosciences*, vol. 7, p. 132, 2020.

[35] G. A. Tribello and P. Gasparotto, "Using dimensionality reduction to analyze protein trajectories," *Frontiers in molecular biosciences*, vol. 6, p. 46, 2019.

[36] M. Duan, J. Fan, M. Li, L. Han, and S. Huo, "Evaluation of dimensionality-reduction methods from peptide folding–unfolding simulations," *Journal of chemical theory and computation*, vol. 9, no. 5, pp. 2490–2497, 2013.

[37] B. Keller, X. Daura, and W. F. Van Gunsteren, "Comparing geometric and kinetic cluster algorithms for molecular simulation data," *The Journal of chemical physics*, vol. 132, no. 7, p. 02B610, 2010.

[38] F. Sittel and G. Stock, "Robust density-based clustering to identify metastable conformational states of proteins," *Journal of Chemical Theory and Computation*, vol. 12, no. 5, pp. 2426–2435, 2016, pMID: 27058020. [Online]. Available: https://doi.org/10.1021/acs.jctc.5b01233

[39] J. D. Chodera, N. Singhal, V. S. Pande, K. A. Dill, and W. C. Swope, "Automatic discovery of metastable states for the construction of markov models of macromolecular conformational dynamics," *The Journal of chemical physics*, vol. 126, no. 15, p. 04B616, 2007.

[40] G. R. Bowman, V. S. Pande, and F. Noé, *An introduction to Markov state models and their application to long timescale molecular simulation*. Springer Science & Business Media, 2013, vol. 797.

[41] P. Novelli, L. Bonati, M. Pontil, and M. Parrinello, "Characterizing metastable states with the help of machine learning," *Journal of Chemical Theory and Computation*, vol. 18, no. 9, pp. 5195–5202, 2022.

[42] S. Koulgi, A. Achalere, U. Sonavane, and R. Joshi, "Markov state modeling analysis captures changes in the temperature-sensitive n-terminal and β-turn regions of the p53 dna-binding domain," *Journal of Chemical Information and Modeling*, vol. 62, no. 24, pp. 6449–6461, 2022.

[43] E. Kots, C. Mlynarczyk, A. Melnick, and G. Khelashvili, "Conformational transitions in btg1 antiproliferative protein and their modulation by disease mutants," *Biophysical Journal*, vol. 121, no. 19, pp. 3753–3764, 2022.

[44] W. Wang, S. Cao, L. Zhu, and X. Huang, "Constructing markov state models to elucidate the functional conformational changes of complex biomolecules," *Wiley Interdisciplinary Reviews: Computational Molecular Science*, vol. 8, no. 1, p. e1343, 2018.

[45] J. D. Chodera and F. Noé, "Markov state models of biomolecular conformational dynamics," *Current opinion in structural biology*, vol. 25, pp. 135–144, 2014.

[46] H. Zhou, Z. Dong, and P. Tao, "Recognition of protein allosteric states and residues: Machine learning approaches," *Journal of computational chemistry*, vol. 39, no. 20, pp. 1481–1490, 2018.

[47] H. S. Hayatshahi, E. Ahuactzin, P. Tao, S. Wang, and J. Liu, "Probing protein allostery as a residue-specific concept via residue response maps," *Journal of Chemical Information and Modeling*, vol. 59, no. 11, pp. 4691–4705, 2019.

[48] C. Wehmeyer and F. Noé, "Time-lagged autoencoders: Deep learning of slow collective variables for molecular kinetics," *The Journal of chemical physics*, vol. 148, no. 24, p. 241703, 2018.

[49] O. Fleetwood, M. A. Kasimova, A. M. Westerlund, and L. Delemotte, "Molecular insights from conformational ensembles via machine learning," *Biophysical Journal*, vol. 118, no. 3, pp. 765–780, 2020.

[50] W. Humphrey, A. Dalke, and K. Schulten, "Vmd: visual molecular dynamics," *Journal of molecular graphics*, vol. 14, no. 1, pp. 33–38, 1996.

[51] J. E. Stone, P. Messmer, R. Sisneros, and K. Schulten, "High performance molecular visualization: In-situ and parallel rendering with egl," in *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, 2016, pp. 1014–1023.

[52] T. Tu, C. A. Rendleman, D. W. Borhani, R. O. Dror, J. Gullingsrud, M. O. Jensen, J. L. Klepeis, P. Maragakis, P. Miller, K. A. Stafford *et al.*, "A scalable parallel framework for analyzing terascale molecular dynamics simulation trajectories," in *SC'08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing*. IEEE, 2008, pp. 1–12.

[53] D. Etezadi, J. B. Warner IV, H. A. Lashuel, and H. Altug, "Real-time in situ secondary structure analysis of protein monolayer with mid-infrared plasmonic nanoantennas," *ACS sensors*, vol. 3, no. 6, pp. 1109–1117, 2018.

[54] R. T. McGibbon, K. A. Beauchamp, M. P. Harrigan, C. Klein, J. M. Swails, C. X. Hernández, C. R. Schwantes, L.-P. Wang, T. J. Lane, and V. S. Pande, "Mdtraj: a modern open library for the analysis of molecular dynamics trajectories," *Biophysical journal*, vol. 109, no. 8, pp. 1528–1532, 2015.

[55] B. Pullman, E. D. Bergmann *et al.*, *Conformation of biological molecules and polymers; proceedings of an international symposium held in Jerusalem, 3-9 April 1972. Edited by Ernst D. Bergmann and Bernard Pullman*, 1973.

[56] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[57] C. Boutsidis and E. Gallopoulos, "Svd based initialization: A head start for nonnegative matrix factorization," *Pattern recognition*, vol. 41, no. 4, pp. 1350–1362, 2008.

[58] A. Beygelzimer, S. Kale, and H. Luo, "Optimal and adaptive algorithms for online boosting," in *International Conference on Machine Learning*. PMLR, 2015, pp. 2323–2331.

[59] R. E. Schapire, "Explaining adaboost," *Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik*, pp. 37–52, 2013.

[60] A. A. Efanov, S. A. Ivliev, and A. G. Shagraev, "Welford's algorithm for weighted statistics," in *2021 3rd International Youth Conference on Radio Electronics, Electrical and Power Engineering (REEPE)*. IEEE, 2021, pp. 1–5.

[61] E. Kots, D. M. Shore, and H. Weinstein, "An equilibrium constant ph molecular dynamics method for accurate prediction of ph-dependence in protein systems: Theory and application," *bioRxiv*, pp. 2020–11, 2020.

[62] ——, "Simulation of ph-dependent conformational transitions in membrane proteins: The clc-ec1 cl-/h+ antiporter," *Molecules*, vol. 26, no. 22, p. 6956, 2021.

[63] G. Morra, A. M. Razavi, A. K. Menon, and G. Khelashvili, "Cholesterol occupies the lipid translocation pathway to block phospholipid scrambling by a g protein-coupled receptor," *Structure*, vol. 30, no. 8, pp. 1208–1217, 2022.

[64] M. Taufer, S. Thomas, M. Wyatt, T. M. Anh Do, L. Pottier, R. F. da Silva, H. Weinstein, M. A. Cuendet, T. Estrada, and E. Deelman, "Characterizing in situ and in transit analytics of molecular dynamics simulations for next-generation supercomputers," in *2019 15th International Conference on eScience (eScience)*, 2019, pp. 188–198.