Parallel Computing: Technology Trends
I. Foster et al. (Eds.)

doi:10.3233/APC200023

Deep Generative Model Driven Protein Folding Simulations

Heng MA [a] Debsindhu BHOWMIK [a] Hyungro LEE [b] Matteo TURILLI [b]
Michael YOUNG [a] Shantenu JHA [b,c] Arvind RAMANATHAN [d]
[a] *CSED, Oak Ridge National Laboratory, Oak Ridge, TN 37830*
[b] *RADICAL, ECE, Rutgers University, Piscataway, NJ 08854, USA*
[c] *Brookhaven National Laboratory, Upton, New York, 11973*
[d] *Data Science and Learning, Argonne National Laboratory, Lemont, IL 60439*

Abstract. Significant progress in computer hardware and software have enabled molecular dynamics (MD) simulations to model complex biological phenomena such as protein folding. However, enabling MD simulations to access biologically relevant timescales (e.g., beyond milliseconds) still remains challenging. These limitations include (1) quantifying which set of states have already been (sufficiently) sampled in an ensemble of MD runs, and (2) identifying novel states from which simulations can be initiated to sample rare events (e.g., sampling folding events). With the recent success of deep learning and artificial intelligence techniques in analyzing large datasets, we posit that these techniques can also be used to adaptively guide MD simulations to model such complex biological phenomena. Leveraging our recently developed unsupervised deep learning technique to cluster protein folding trajectories into partially folded intermediates, we build an iterative workflow that enables our generative model to be coupled with all-atom MD simulations to fold small protein systems on emerging high performance computing platforms. We demonstrate our approach in folding Fs-peptide and the β-β-α (BBA) fold, FSD-EY. Our adaptive workflow enables us to achieve an overall root-mean squared deviation (RMSD) to the native state of 1.6 $\AA$ and 4.4 $\AA$ respectively for Fs-peptide and FSD-EY. We also highlight some emerging challenges in the context of designing scalable workflows when data intensive deep learning techniques are coupled to compute intensive MD simulations.

Keywords. Deep learning, Workflows, Molecular dynamics, Protein folding

1. Introduction

Multiscale molecular simulations are widely used to model complex biological phenomena, such as protein folding, protein-ligand (e.g., small molecule, ligand/ drug, protein) interactions, and self-assembly [1,2]. However, much of these phenomena occur at timescales that are fundamentally challenging for molecular simulations to access, even with advances in both hardware and software technologies [3]. Hence, there is a need to develop scalable, adaptive simulation strategies that can enable sampling of timescales relevant to these biological phenomena.

Many adaptive sampling techniques [4,5,6,7,8,9] have been proposed. All these techniques share some similar characteristics, including (a) the need for efficient and automated approaches to identify a small number of relevant conformational coordinates

(either through clustering and/or dimensionality reduction techniques) [10,11,12], and (b) the identification of the 'next' set of simulations to run such that more trajectories are successful in attaining a specific end goal (e.g., protein that is well folded, protein bound to its target ligand, etc.) [8,9].

These adaptive simulations present methodological and infrastructral challenges. Ref. [4] provides important validation of the power of adaptive methods over traditional "vanilla" molecular dynamics (MD) simulations or "ensemble" simulations. Ref. [13] highlights challenges of such workflows on high-performance computing platforms.

We recently developed a deep learning based approach that uses convolutions and a variational autoencoder (CVAE) to cluster simulations in an unsupervised manner [14]. We have shown that our CVAE can discover and identify intermediate states from protein folding pathways; further, the CVAE-learned latent dimensions cluster conformations into biophysically relevant features such as number of native contacts, or root mean squared deviation (RMSD) to the native state.

We posit that the CVAE learned latent features can be used to drive adaptive sampling within MD simulations, where the next set of simulations to run are decided based on a measure of 'novelty' of the simulation/ trajectory frame observed.

Integrating CVAE concurrently with large-scale ensemble simulations on high-peformance computing platforms entails the aforementioned complexity of adaptive workflows [13], while introducing additional infrastructural challenges. These arise from the concurrent and adaptive execution of heterogeneous simulations and learning workloads requiring sophisticated workload and performance balancing, inter alia.

In this paper, we implement a baseline version of our deep learning driven adaptive sampling workflow with multiple concurrent instances of MD simulations and CVAEs. Our contributions can be summarized as follows:

- We demonstrate that deep learning based approaches can be used to drive adaptive MD simulations at scale. We demonstrate our approach in folding small proteins, namely Fs-peptide and the β-β-α-fold (BBA) protein and show that it is possible to fold them using deep learning driven adaptive sampling strategy.
- We highlight parallel computing challenges arising from the unique characteristics of the worklfow, viz., training of deep learning algorithms can take almost as much time as running simulations, necessitating novel developments to deal with heterogeneous task placement, resource management and scheduling.

Taken together, our approach demonstrates the feasibility of coupling deep learning (DL) and artificial intelligence (AI) workflows with conventional all-atom MD simulations.

2. Related Work

Adaptive sampling techniques have been widely developed for MD simulations. A thorough review of this area of research is beyond the scope of this paper – however, we refer the interested reader to [15] for more details. From a computational point of view, adaptive sampling techniques require the use of specialized middle-ware that allows for scheduling, managing and orchestrating hundreds (if not thousands) of loosely coupled simulations that are guided by statistical approaches [16,17,18].

More recently, the development of machine learning techniques such as Markov State Models (MSM) and its integration with adaptive sampling techniques have proven quite useful for selecting the optimal number of stating points (for simulations) while simultaneously improving the convergence in these simulations [19]. When combined with sampling techniques such as umbrella sampling, MSM-based estimators can provide further insights into complex biological processes [20]. With advances in machine learning techniques, especially deep learning methods [21,14,22,8], we examined whether using such generative models could be advantageous in the context of accelerating protein folding simulations.

3. Methods

3.1. Workflow description

Two key components of the workflow include the MD simulation module and the deep-learning based CVAE module, which are described below.

Molecular dynamics (MD) simulations: The MD simulations are performed on GPUs with OpenMM 7.3.0 [23]. Both the Fs-peptide and BBA systems were modeled using the Amberff99SB-ildn force field [24] in implicit Onufriev-Bashford-Case GBSA solvent model [25]. The non-bonded interactions are cut off at 10.0 $\mathring{A}$ and no periodic boundary condition was applied. All the bonds to hydrogen are fixed to their equilibrium value and simulations were run using a 2 fs time step. Langevin integrator was used to maintain the system temperature at 300 K with a friction coefficient at 91 ps^{-1}. The initial configuration was optimized using L-BFGS local energy minimizer with tolerance of 10 kJ/-mol and maximum of 100 iterations. The initial velocity is assigned to each atom from a Boltzmann distribution at 300 K. We also added a new reporter to calculate the contact matrix of C_α atoms in the protein (using a distance cut-off of 8 $\mathring{A}$ in hdf5 format using the MDAnalysis module [26,27] that could be used as inputs to the deep learning module (described below). Each simulation run outputs a frame every 50 ps.

Convolutional Variational Autoencoder (CVAE): Autoencoder is a deep neural network architecture that can represent high dimensional data in a low dimensional latent space while retaining the key information [28]. With its unique hourglass shaped architecture, an autoencoder compresses input data into a latent space with reduced dimension and reconstructs it to the original data. We use the CVAE to cluster the conformations from our simulations in an unsupervised manner [14,29]. Currently in our workflow, we use the number of latent dimensions as a hyperparameter (varying between $\{3,\ldots,6\}$) and use the CVAE that most accurately reconstructs the input contact maps [14,29]. CVAE was implemented using Keras/TensorFlow and trained on a V100 GPU for 100 epochs.

Assembling our workflow: As illustrated in Figure 1, our prototype workflow couples the two components. In the first stage, the objective is to initially train the CVAE to determine the optimal number of latent dimensions required to faithfully reconstruct the simulation data. We commence our runs as an ensemble of equilibrium MD simulations. Ensemble MD simulations are known to enable better sampling of the conformational landscape, and also can be run in an embarrassingly parallel manner. The simulation data is converted into a contact map representation (to overcome issues with rotation/translation within the simulation box) and are streamed at regular intervals into the CVAE module. The output from the first stage is an optimally learned latent representation of the simu-

lation data, which organizes the landscape into clusters consisting of conformations with similar biophysical features (e.g., RMSD to the native state). Note that this is an emergent property of the clustering and the RMSD to the native state is not used as part of training data.

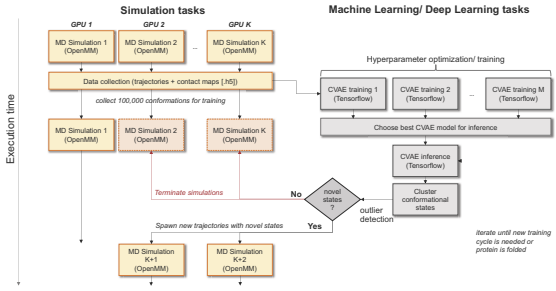

Figure 1.: Deep generative model driven protein folding simulation workflow.

In the second stage, our objective is to identify the most viable/ promising next set of starting states for propagating our MD simulations towards the folded state. We switch the use of CVAE to infer from newly generated contact maps (from simulations) and observe how they are clustered. Based on their similarity to the native state (measured by the RMSD), a subset of these conformations are selected for propagating additional MD runs. The workflow is continued until the protein is folded (i.e., conformations reach a user-defined RMSD value to the native state).

3.2. Implementation, Software and Compute Platform

We used the Celery software to implement the aforementioned workflow. Celery is an asynchronous task scheduler with a flexible distributed system to process messages and manage operations, which enables real-time task processing and scheduling. The tasks can be executed and controlled by the Celery worker server asynchronously or synchronously. Celery applications use callables to represent the modules that are part of the workflow. Once called, the task client adds to the task queue a message where its unique name is referred so that the worker can locate the right function to execute. The flexibility of Celery framework enables real-time interfacing to manage resource and excise control over the task scheduling and execution. With MD simulation and CVAE tasks modulized as Celery compatible callables, they are monitored and controlled through Celery Python interface. According to the strategy demonstrated in Figure 1, we simply build multi-task workflows, which supports a large volume of concurrent tasks with real-time interfacing and decision-making. The use of Celery framework allows us to establish a baseline for estimating the compute requirements of our workflow.

We tested our deep learning driven adaptive simulation framework on the NVIDIA DGX-2 system at Oak Ridge National Laboratory (ORNL). The DGX-2 system provides more than 2 petaflops of computational power from a single node that leverages its 16 interconnected NVIDIA Tesla V100-SXM3-32GB GPUs. This enables us to distribute the MD simulations and CVAE training onto 12 and 4 GPUs respectively. All the components in the workflow are encapsulated within a Python script that manages the various tasks through Celery. It first initializes the Celery worker along with the selected broker, RabbitMQ. All 16 GPUs are then employed for MD simulations to first generate 100,000 conformers as the initial training data for CVAE. With 5 minute interval be-

tween iterations, the trained CVAE periodically compress MD simulation conformers from MD trajectories into data points of latent space, which are subsequently evaluated with density-based spatial clustering of applications with noise (DBSCAN) for outlying conformations [30]. We used DBSCAN for its relative simplicity and also to establish a baseline implementation of our code. For Fs-peptide, outliers were collected with all four trained CVAE models and only CVAE with 6 dimensional latent space was applied for BBA outlier searching. In each iteration, the MD runs are examined for outliers. Simulations that pass an initial threshold of 20,000 frames (1 μs) for Fs-peptide and 10,000 (0.5 μs) for BBA, but do not produce any outliers for the last 5000 frames (250 ns of simulation time) are purged. With the available GPUs from such MD runs, new MD simulations are launched from the the outliers to ensure appropriate resource management and usage.

4. Results

In previous work [14], we have shown that the CVAE can learn a latent space from the Fs-peptide simulations such that the conformations from the simulations cluster into distinct clusters consisting of folded and unfolded states. When parameters such as the RMSD (to the native state) and the fraction of native contacts are used to annotate the latent dimensions [31], we showed that these latent representations correspond to reaction coordinates that describe how a protein may fold (beginning with the unfolded state ensemble). Thus, we posit that we can propagate the simulations along these low-dimensional representations and can drive simulations to sample folded states of the protein in a relatively short number of iterations.

Figure 2 summarizes the results of our folding simulations of Fs-peptide. The peptide consists of 21 residues – Ace-A_5(AAARA)$_3$A-NME – where Ace and NME represent the N- and C-terminal end caps of the peptide respectively, and A represents the amino acid Alanine, where as R represents the amino acid Arginine. It is often used as a prototypical system for protein folding and adopts a fully helical structure as part of its native state ensemble [32]. Previous simulations used implicit solvent simulations using the GBSA-OBC potentials and the AMBER-FF99SB-ILDN force-field with an aggregate simulation time of 14 μs at 300K [32]. We used the same settings for our MD simulations and initiated our workflow. Summary statistics of the simulations are provided in Table 1. A total of 90 iterations of the workflow was run to obtain a total sampling of 54.198 μs. Note that the sampling time of the MD simulations is an aggregate measure similar to the ones reported in previous studies.

We began by examining the RMSD with respect to the native state from all of our simulations. As shown in Figure 2A, 13 of the total of 31 simulations are unproductive – i.e., they do not sample the native state consisting of the fully formed α-helix. This is not entirely surprising given that the starting state consists of a nearly linear peptide with no residual secondary structures. Based on this observation, we posited that our CVAE model can be used to identify partially folded states from the simulations. We also examined the histogram of the RMSD values computed for each conformation with respect to the native state ensemble (Figure 2B). Based on the histograms, we can reasonably choose a threshold of 3.1$\mathring{A}$ or less to depict the folded state ensemble, followed by 4.6 $\mathring{A}$ for partially folded states, and 8.3 $\mathring{A}$ for the unfolded states. Any trajectory that shows RMSD values beyond 8.3 $\mathring{A}$ are only sampling the unfolded state of the protein.

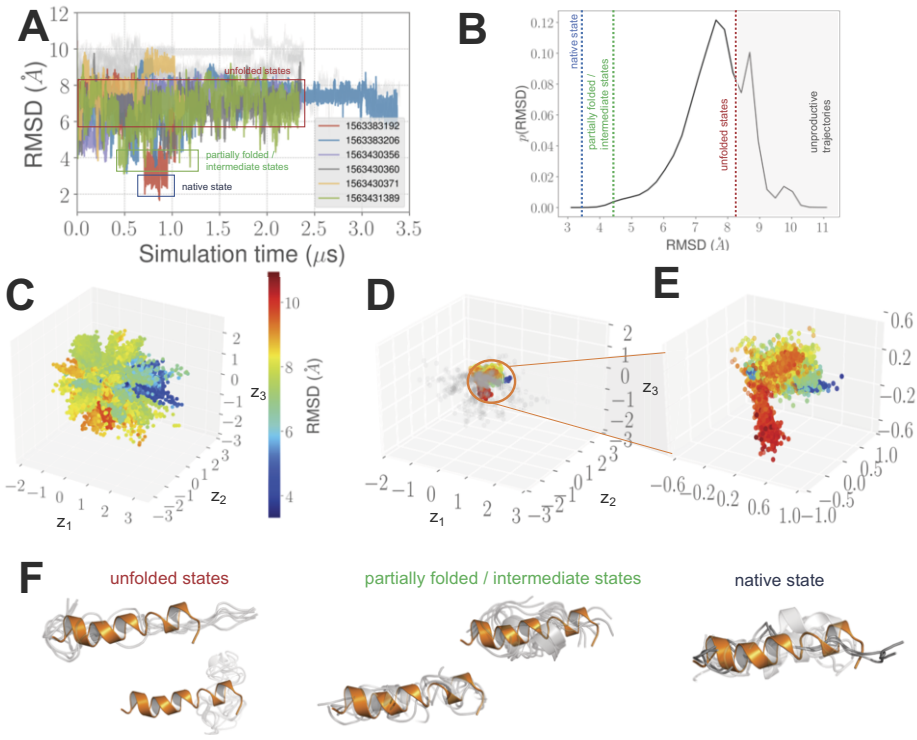

Figure 2. CVAE-driven folding simulations of Fs-peptide.(A) Root mean squared deviation (RMSD) with respect to the native/ folded state from the 31 trajectories generated using our adaptive workflow for the Fs-peptide system. Only productive simulations – i.e., simulations that achieve a RMSD cut-off of 4.5 $\AA$ or less are highlighted for clarity. The rest of the simulations are shown in light gray. (B) A histogram of the RMSD values in panel (A) depicting the RMSD cut-off for identifying folded, partially folded, and unfolded ensembles from the data. The corresponding regions are also marked in panel (A). (C) Using the RMSD to the native state as a measure of foldedness of the system, we project the simulation data onto a three dimensional latent representation learned by the CVAE. Note that the folded states (low RMSD values highlighted in deeper shades of blue) are separated from the folding intermediate (shades of green and yellow) and the unfolded states (darker shades of red).(D) A zoomed in projection of the last 0.5 μs of simulations generated along with the original projections (shown in pale gray, subsampled at every 100^{th} snapshot). (E) highlights the same but just showing the samples from the last 0.5 μs to highlight the differences between folded and unfolded states. (F) shows representative snapshots from our simulations with respect to the unfolded, partial folded, and native state ensembles. Note that the cartoon representation shown in orange represents the native state (minimum RMSD of 1.6 $\AA$ to reference structure) determined from our simulations.

The projections of all the 31 simulations onto the learned CVAE is depicted in Figure 2C. Collectively, z_1-z_3 provide a description of the Fs-peptide folding process. Notably, much of the folded conformational states (highlighted in blue, indicating low RMSD to the native state) are clustered together. Similarly, the unfolded conformations (conformations colored in darker shades of red with higher RMSD to the native state ensemble) are also clustered together. Taking this further, we examined if the similarity in the conformations hold even with a smaller partition of the data (see Figures 2D and E), namely the last 10% of the overall simulation data. This can be treated as a test set from which new simulations are initiated. Notably, from these simulations we observe the presence of roughly three arms in the projections (Figure 2E) consisting of: (1) partially folded

System	Total no. simulations	Total simulation time (μs)	(Shortest*, Longest) simulations (μs)	Iterations	Min. RMSD ($\AA$)
Fs-peptide	31	54.198	1.01, 3.4	90	1.6
BBA (FSD-EY)	45	18.562	0.517, 0.873	100	4.44

Table 1. Summary statistics of simulations. *Only considering the simulations that pass the initial threshold.

highlighted in shades of green/yellow, (2) unfolded state ensemble highlighted in shades of red, and (3) a much smaller ensemble of folded states (highlighted in blue).

For each of these states, we can also extract the structural characteristics with respect to the folded state (Figure 2F). Many of the unfolded states do not consist of any secondary structural features (top and bottom left panels). The partially folded states consist of partial turns/ helical structures. The final folded state (with RMSD of 1.6 $\AA$) consists of most (if not all) helical turns in the protein.

4.1. Folding simulations of FSD-EY

The BBA protein namely, FSD-EY is a designed protein that adopts a β-β-α-fold in its native state; however this protein tends to be dynamic in solution [33,34]. Similar to other zinc-finger proteins, the structure of the protein can potentially vary, and represents a challenging use-case for testing our workflow. As shown in Figure. 3, our simulations do start with a completly unfolded state of the protein (average RMSD to native state is about 12 $\AA$. Using an aggregated MD sampling time of 18 μs, we note that we reach a RMSD value of 4.44 $\AA$.

Although we do not sample the native state of the protein consisting of the β-β-α-fold, we are still able to sample regions of the landscape that consist of a defined hydrophobic core consisting of the highlighted residues in Figure 3D. Except for the dynamic C-terminal end, where the hydrophobic interactions between F21 and F25 are not entirely stable, the conformations that exhibit low RMSD values to the native state depict the presence of this hydrophobic core. We expect that extending these simulations further using the CVAE-driven protocol will enhance these interactions allowing it to fold completely.

5. Discussion

As artificial intelligence (AI) and deep learning (DL) techniques become more pervasive for analyzing scientific datasets, there is an emerging need for supporting AI/DL coupled workflows to traditional HPC applications such as MD simulations. Our approach provides a proof-of-concept for how we can guide MD simulations to sample folded state ensemble of small proteins using DL techniques. The approach that we chose was based on building a generative model for protein conformations and identifying new starting conformations for additional MD sampling. Although the generative model was only used to identify novel conformations for extending our MD simulations, it nevertheless allowed us to guide the MD simulations towards sampling folded conformations of the protein systems we considered.

Although DL approaches can take significantly longer time to train, we deliberately chose a prototypic DL approach, namely CVAE, to train on our MD simulation data (Ta-

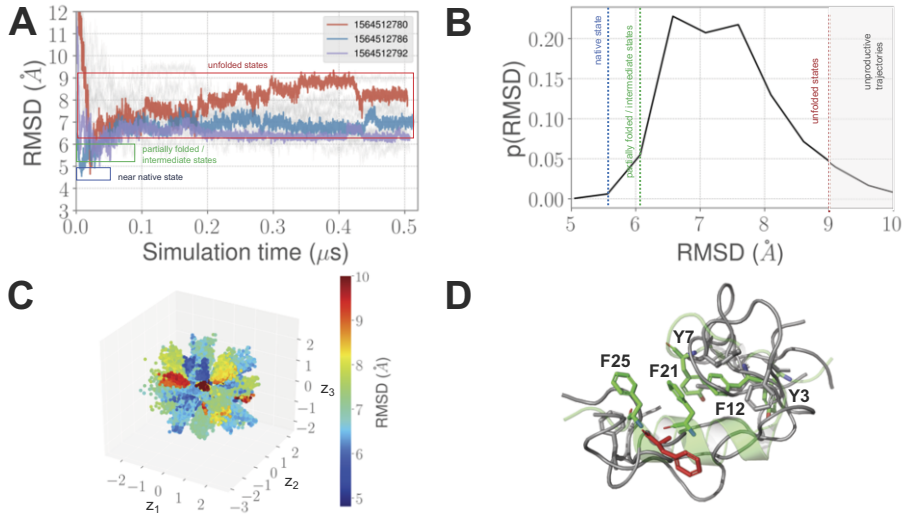

Figure 3. CVAE-driven folding simulations of BBA-fold, FSD-EY. (A) RMSD plots with respect to the native state of FSD-EY depicting the near-native state (blue), partially folded states (green) and unfolded (red) trajectories similar to Figure 2. (B) A histogram of the RSMD values to the native state. (C) The learned projections from the CVAE for the trajecotries; similar to the Fs-peptide system, we can observe the clustering of conformations based on their RMSD values to the native state. We have used a RMSD cut-off of 10Å to highlight states closer to the native state. (D) Although we could not fully fold the protein, we do observe the presence of a well-formed hydrophobic core except for one residue (F25) at the C-terminal end of the protein.

System	DL training (100 epochs; minutes)	Time per epoch (seconds)	Inference time (ms/frame)	MD simulations (ns per minute)
Fs-peptide	7	5	5.13	1.25
BBA	11	7	1.27	1.20

Table 2. Summary statistics of time taken by the individual components of our workflow: (1) train and infer from the CVAE for each system, and (2) running the MD simulation.

ble 2). As can be seen from the table, the computational cost of training and inference times for the CVAE model is on par with the cost for running our MD simulations. That is, within the time required to train our CVAE model, our MD simulations progress only by about a nanosecond. Thus, starting up of new MD simulations based on the guidance received from our CVAE model will not affect the workflow's overall performance. Further, our MD simulations were run using implicit solvent models, which also significantly reduces their computational times. Further, each contact map is no more than a couple of kilobytes of data and hence we did not require the utilization of intrinsic capabilities of the NVIDIA DGX-2, including the ability to potentially stream data across GPUs/ processors.

A primary motivation for this work was to use ML/DL based analysis to drive MD simulations, and to calibrate results against non ML/DL driven approaches. In Ref. [12] Fox et al introduced the concept of "Effective Performance" that is achieved by combining learning with simulation and without changing the traditional system characteristics. Our selection of physical systems, in particular the BBA peptide allows to provide a coarse-grained estimate for the effective performance of CVAE based adaptive sam-

pling. Using Ref. [4] as reference data, we find that the effective performance of CVAE based sampling is at least a factor of 20 greater than "vanilla" MSM based sampling approaches. Our estimate is based upon the convergence of simulated BBA structures to its reference structures to within 4.5 Å. Note that the ExTASY based simulations in Ref. [4] are at least two orders of magnitude more efficient that reference DE Shaw simulations. In future work, we will extend our effective performance estimate to Villin head piece (VHP) and refine our estimates for BBA.

We expect that the concomitant increase in data sizes for larger MD simulations would necessitate the use of streaming approaches. Similarly, as the time required to train our DL models with data-/model- parallel approaches increases, it will require the use of emerging memory hierarchy architectures to facilitate efficient data handling/ transfer across compute nodes that are dedicated for training and simulation. Further, data intensive techniques such as reinforcement learning and/or active learning could also have been used to guide our MD simulations.

The requirements of the ML/DL driven simulations outlined in this paper are representative of ML/DL driven adaptive workflows — where the status of the intermediate data analysis drive subsequent computations. Adaptive workflows pose significant challenges [13] compared to workflows whose execution trajectory is predetermined a priori. Further, the integration of diverse ML/DL approaches as the intermediate analysis driving subsequent computations adds additional complexity, which includes but is not limited to heterogeneous workload, load balancing and resource management. Scalable execution and modern HPC platforms implies the need for specialized middleware that support address these challenges. We are addressing these aspects as part of ongoing work and future development built upon RADICAL-Cybertools [13,35].

The source code, associated datasets including the generated simulations and deep learning models are available on https://github.com/acadev/Parco2019_baseline.

Acknowledgements

We thank Vivek Balasubramanian and Jumana Dakka for helpful discussions and early contributions. We also thank Chris Layton for helping set up our runs on the NVIDIA DGX-2 compute systems. We also acknowledge support by NSF DIBBS 1443054 and NSF RADICAL-Cybertools 1440677. This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of the manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan). This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05- 00OR22725.

References

[1] Ron O. Dror, Robert M. Dirks, J.P. Grossman, Huafeng Xu, and David E. Shaw. Biomolecular simulation: a computational microscope for molecular biology. *Annu Rev Biophys*, 41(1):429–452, 2012.

[2] Eric H. Lee, Jen Hsin, Marcos Sotomayor, Gemma Comellas, and Klaus Schulten. Discovery through the computational microscope. *Structure*, 17(10):1295–1306.

[3] Gregory R. Bowman, Kyle A. Beauchamp, George Boxer, and Vijay S. Pande. Progress and challenges in the automated construction of markov state models for full protein systems. *J Chem Phys*, 131(12):124101, 2009.

[4] Eugen Hruska, Vivekanandan Balasubramanian, John R Ossyra, Shantenu Jha, and Cecilia Clementi. Extensible and scalable adaptive sampling on supercomputers. *arXiv preprint arXiv:1907.06954*, 2019.

[5] Nina Singhal Hinrichs and Vijay S. Pande. Calculation of the distribution of eigenvalues and eigenvectors in markovian state models for molecular dynamics. *The Journal of Chemical Physics*, 126(24):244101, 2007.

[6] Jeffrey K. Weber and Vijay S. Pande. Characterization and rapid sampling of protein folding markov state model topologies. *J Chem Theory Computat*, 7(10):3405–3411, 2011. PMID: 22140370.

[7] S. Doerr, I. Ariz-Extreme, M. J. Harvey, and G. De Fabritiis. Dimensionality reduction methods for molecular simulations. *ArXiv e-prints*, October 2017.

[8] Shriyaa Mittal and Diwakar Shukla. Recruiting machine learning methods for molecular simulations of proteins. *Molecular Simulation*, 44(11):891–904, 2018.

[9] Zahra Shamsi, Kevin J. Cheng, and Diwakar Shukla. Reinforcement learning based adaptive sampling: Reaping rewards by exploring protein conformational landscapes. *The Journal of Physical Chemistry B*, 122(35):8386–8395, 09 2018.

[10] Michael R. Shirts and Vijay S. Pande. Mathematical analysis of coupled parallel simulations. *Phys Rev Lett*, 86(22):4983–4987, May 2001.

[11] A. J. Savol, V. M. Burger, P. K. Agarwal, A. Ramanathan, and C. S. Chennubhotla. QAARM: quasi-anharmonic autoregressive model reveals molecular recognition pathways in ubiquitin. *Bioinformatics*, 27(13):52–60, Jul 2011.

[12] Geoffrey Fox, James A Glazier, JCS Kadupitiya, Vikram Jadhao, Minje Kim, Judy Qiu, James P Sluka, Endre Somogyi, Madhav Marathe, Abhijin Adiga, and Shantenu Jha. Learning everywhere: Pervasive machine learning for effective high-performance computation. *High-Performance Big Data Computing, IPDPS Workshop, Rio de Janeiro*, 2019.

[13] Vivek Balasubramanian, Travis Jensen, Matteo Turilli, Peter M. Kasson, Michael R. Shirts, and Shantenu Jha. Implementing adaptive ensemble biomolecular applications at scale. *CoRR*, abs/1804.04736, 2018.

[14] Debsindhu Bhowmik, Shang Gao, Michael T. Young, and Arvind Ramanathan. Deep clustering of protein folding simulations. *BMC Bioinformatics*, 19(18):484, 2018.

[15] Peter M Kasson and Shantenu Jha. Adaptive ensemble simulations of biomolecules. *Current Opinion in Structural Biology*, 52:87 – 94, 2018. Cryo electron microscopy: the impact of the cryo-EM revolution in biology • Biophysical and computational methods - Part A.

[16] Sander Pronk, Per Larsson, Iman Pouya, Gregory R. Bowman, Imran S. Haque, Kyle Beauchamp, Berk Hess, Vijay S. Pande, Peter M. Kasson, and Erik Lindahl. Copernicus: A new paradigm for parallel adaptive molecular dynamics. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, pages 60:1–60:10, New York, NY, USA, 2011. ACM.

[17] Vivek Balasubramanian, Travis Jensen, Matteo Turilli, Peter Kasson, Michael Shirts, and Shantenu Jha. Implementing adaptive ensemble biomolecular applications at scale. *arXiv preprint arXiv:1804.04736*, 15:800–809, 2018.

[18] Matthew C. Zwier, Joshua L. Adelman, Joseph W. Kaus, Adam J. Pratt, Kim F. Wong, Nicholas B. Rego, Ernesto Suárez, Steven Lettieri, David W. Wang, Michael Grabe, Daniel M. Zuckerman, and Lillian T. Chong. Westpa: An interoperable, highly scalable software package for weighted ensemble simulation and analysis. *Journal of Chemical Theory and Computation*, 11(2):800–809, 2015. PMID: 26392815.

[19] Nuria Plattner, Stefan Doerr, Gianni De Fabritiis, and Frank Noé. Complete protein–protein association kinetics in atomic detail revealed by molecular dynamics simulations and markov modelling. *Nature Chemistry*, 9(10):1005–1011, 2017.

[20] Sunhwan Jo, Donghyuk Suh, Ziwei He, Christophe Chipot, and Benoît Roux. Leveraging the information from markov state models to improve the convergence of umbrella sampling simulations. *The*

Journal of Physical Chemistry B, 120(33):8733–8742, 08 2016.

[21] Andreas Mardt, Luca Pasquali, Hao Wu, and Frank Noé. Vampnets for deep learning of molecular kinetics. *Nature Communications*, 9(1):5, 2018.

[22] Frank Noé, Alexandre Tkatchenko, Klaus-Robert Müller, and Cecilia Clementi. Machine learning for molecular simulation. *arXiv preprint arXiv:1911.02792*, 2019.

[23] Peter Eastman, Jason Swails, John D. Chodera, Robert T. McGibbon, Yutong Zhao, Kyle A. Beauchamp, Lee-Ping Wang, Andrew C. Simmonett, Matthew P. Harrigan, Chaya D. Stern, Rafal P. Wiewiora, Bernard R. Brooks, and Vijay S. Pande. Openmm 7: Rapid development of high performance algorithms for molecular dynamics. *PLOS Computational Biology*, 13(7):1–17, 07 2017.

[24] Kresten Lindorff-Larsen, Stefano Piana, Ron O. Dror, and David E. Shaw. How fast-folding proteins fold. *Science*, 334(6055):517–520, 2011.

[25] Alexey Onufriev, Donald Bashford, and David A. Case. Exploring protein native states and large-scale conformational changes with a modified generalized born model. *Proteins: Structure, Function, and Bioinformatics*, 55(2):383–394, 2004.

[26] Naveen Michaud-Agrawal, Elizabeth J. Denning, Thomas B. Woolf, and Oliver Beckstein. Mdanalysis: A toolkit for the analysis of molecular dynamics simulations. *J Comput Chem*, 32(10), 2011.

[27] R. J. Gowers, M. Linke, Jonathan Barnoud, Tyler J. E. Reddy, Manuel N. Melo, Sean L. Seyler, Jan Domanski, David L. Dotson, Sebastien Buchoux, Ian M. Kenney, and Oliver Beckstein. MDAnalysis: A python package for the rapid analysis of molecular dynamics simulations. In Sebastian Benthall and Scott Rostrup, editors, *Proceedings of the 15th Python in Science Conference*, pages 98 – 105, 2016.

[28] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[29] Raquel Romero, Arvind Ramanathan, Tony Yuen, Debsindhu Bhowmik, Mehr Mathew, Lubna Bashir Munshi, Seher Javaid, Madison Bloch, Daria Lizneva, Alina Rahimova, Ayesha Khan, Charit Taneja, Se-Min Kim, Li Sun, Maria I. New, Shozeb Haider, and Mone Zaidi. Mechanism of glucocerebrosidase activation and dysfunction in gaucher disease unraveled by molecular dynamics and deep learning. *Proceedings of the National Academy of Sciences*, 116(11):5086–5095, 2019.

[30] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.

[31] Jörg Gsponer and Amedeo Caflisch. Molecular dynamics simulations of protein folding from the transition state. *Proceedings of the National Academy of Sciences*, 99(10):6719–6724, 2002.

[32] Robert T. McGibbon. Fs MD Trajectories. 5 2014.

[33] Kresten Lindorff-Larsen, Paul Maragakis, Stefano Piana, Michael P. Eastwood, Ron O. Dror, and David E. Shaw. Systematic validation of protein force fields against experimental data. *PLOS ONE*, 7(2):e32131–, 02 2012.

[34] Catherine A Sarisky and Stephen L Mayo. The $\beta\beta\alpha$-fold: explorations in sequence space. *Journal of Molecular Biology*, 307(5):1411 – 1418, 2001.

[35] Matteo Turilli, Vivek Balasubramanian, Andre Merzky, Ioannis Paraskevakos, and Shantenu Jha. Middleware building blocks for workflow systems. *Computing in Science & Engineering (CiSE) special issue on Incorporating Scientific Workflows in Computing Research Processes*, https://arxiv.org/abs/1903.10057.