Benchmarking Machine Learning Workloads in Structural Bioinformatics Applications

Heng Ma, Austin Clyde, Anda Trifan, Venkatram Vishwanath, Arvind Ramanathan
{heng.ma,aclyde,venkat,ramanathana}@anl.gov
Data Science & Learning Division, Argonne National Laboratory
Lemont, Illinois

Debsindhu Bhowmik
bhowmikd@ornl.gov
Computational Science & Engineering Division, Oak Ridge National Laboratory
Oak Ridge, Tennessee

Shantenu Jha
shantenu.jha@rutgers.edu
Rutgers University and Brookhaven National Laboratory
New Brunswick, New Jersey

ABSTRACT

Benchmarking machine learning (ML) based methods have traditionally been largely uncoupled from scientific simulations. However, there has been considerable interest in using learning approaches in the context of scientific simulation software to: (1) analyze large volumes of simulation data (or archived databases); (2) drive adaptive simulations to sample 'rare' events; (3) accelerate simulations by replacing expensive computational kernels with efficient ML inference techniques, and (4) drive optimal simulation strategies based on ML guided approaches. Thus, the coupling of learning with simulation tools can range widely: from ML approaches which are independent of the application itself, to ML approaches which are used to drive large-scale simulations. Using structural bioinformatics applications, we motivate how ML approaches are coupled with physics-based simulations. To optimize such coupled applications on emerging hardware and software platforms, we need to consider additional and often unique performance considerations. In this paper, we present an overview of different learning approaches in structural bioinformatics applications, performance considerations for such coupled applications, and outline the development of performance metrics. We hope this will enable the broader scientific community as well as hardware and software vendors to evaluate the role of learning tools when coupled to scientific simulation applications, and hope that this could serve as a framework for other application domains.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; **Molecular simulation**.

KEYWORDS

structural bioinformatics, artificial intelligence, molecular dynamics, molecular docking, benchmarks

CHALLENGE '20, Austin, TX,

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/1122445.1122456

ACM Reference Format:
Heng Ma, Austin Clyde, Anda Trifan, Venkatram Vishwanath, Arvind Ramanathan, Debsindhu Bhowmik, and Shantenu Jha. 2018. Benchmarking Machine Learning Workloads in Structural Bioinformatics Applications. In *Proceedings of First International Workshop on Benchmarking Machine Learning Workloads on Emerging Hardware (CHALLENGE '20).* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Biological macro-molecules, including de-oxy/ribose nucleic acids (D/RNA), proteins, carbohydrates and lipids are essential workhorses of a living cell. These diverse macro-molecules mediate several essential functions within a cell and are therefore implicated in maintaining cellular health. Consequently, their dysfunction is implicated in cancer, diabetes, cardio-vascular, neuromuscular, and neurological disease conditions. Therefore, understanding how the three-dimensional (3D) structures of such macro-molecules, and their dynamics effect diverse biological functions remains one of the longstanding goals of molecular structural biology. Despite tremendous progress in the last several decades in experimental determination of 3D macro-molecular structure using techniques such as X-ray crystallography, nuclear magnetic resonance, cryo-electron microscopy, X-ray/neutron scattering [29], (i) the sheer gap between the number of available gene/protein sequences and available 3D structures, and (ii) inherent limitations in the time- and length-scales accessible to each of these techniques, make it challenging to build a comprehensive mapping between macro-molecular structure and functions [8].

Computational techniques, on the other hand, have played a vital and complementary role in mapping macro-molecular structure and function [12]. In particular, given that a protein's 3D structure is more conserved than its primary sequence, one can exploit the similarity in protein sequences to determine its most likely 3D structure[1]. This is commonly referred to as the protein structure prediction problem or automated homology modeling (see Fig.1. This problem has been widely addressed in the literature [13], with a number of approaches being developed that include both modeling and simulation techniques as well as ML/DL techniques [25]. Given that no one experimental structure determination technique can access time- and length-scales associated with the biological

[1]In this paper, we present our benchmarks from the perspective of proteomics – we use protein structure determination and analysis as our primary motivation. However, the concepts used here and the proposed benchmarks are widely applicable to other structural bioinformatics techniques.

functions of macromolecules, multiscale biophysical simulation techniques such as molecular dynamics (MD) and Markov chain Monte Carlo (MCMC) have played an important role in providing insights into how motions at the atomistic scale affect biological function(s) [1, 6]. These simulations are governed by a potential energy function that includes atomistic interactions (both bonded and non-bonded terms) and integrate Newton's laws of motion along suitable time steps (typically in the range of femtoseconds/10^{-15}s for all-atom, up to picoseconds/10^{-12}s, depending on the length scales of either all-atom or coarse-grained simulations) to generate trajectories. With improvements in both hardware [28] and software[5, 7], these simulations have been able to simulate complex (and emergent) biological phenomena such as protein folding, ligand binding/unbinding, as well as phase separation in large protein aggregates.

The volume and size of these trajectories entails that ML/DL/AI techniques can be important in: (i) identifying events of interest within these simulations, e.g., protein folding events across millisecond timescale trajectories; (ii) reducing the overall dimensionality of the simulation trajectories to identify order parameters along which the biological phenomenon can be quantified, and (iii) clustering simulation data having underlying biological and physical significance, e.g., quantitatively linking observed conformational transitions in a trajectory with experimental observations [21].

Traditional analysis for long timescale simulation trajectories have largely relied on approaches such as principal component analysis and/or independent component analysis, as well as nonlinear dimensionality reduction techniques such as isoMAP to map the conformational landscapes of proteins and other macromolecules [11, 20, 24, 30, 33]. However, the inherent statistical structure of atomic fluctuations analyzed from long timescale simulations (as well as experimental techniques) have shown that assumptions of orthogonality or linearity do not hold. This motivates the development of deep learning techniques to analyze such datasets. Indeed, deep learning techniques have now been widely developed for analyzing MD simulations [3].

MD techniques often face the challenge of accessing timescales that are related to biological function. In such scenarios, adaptive sampling techniques to enable ensemble MD simulations to access massive parallelism [4] have been used. More recently, ML techniques have also been used to design adaptive sampling methods and have shown promise in accelerating protein folding simulations [14, 17]. Additionally, simulations can also be directed by artificial intelligence (AI) techniques such as reinforcement learning (RL) to model phenomena such as protein-ligand (small molecule) interactions [27, 32].

As shown Fig. 1, ML/DL/AI technqiues used in structural bioinformatics applications are coupled to physical simulations. In particular, starting with: (1) automated approaches to model protein structure using similarities in protein sequences, PDB models that are generated are simulated using MD. The data generated is then (2) clustered into conformational states and then analyzed to (3) identify novel states that can then be used to restart or continue new simulations. This constitutes a typical adaptive sampling run, where clustering and event identification go hand-in-hand to simulate complex biological phenomena. In addition, it is also feasible to consider (4) small molecules and how they interact with each of the states identified in the simulations. RL techniques have played an important role. Finally, (5) while RL techniques provide a candidate list of molecules that look promising (for docking), further analysis and scoring is needed which can be interfaced with molecular modeling and simulations.

Despite the proliferation of a diverse set of ML/DL/AI tools for protein structure determination and MD simulations, characterizing the runtime and performance of these tools on emerging (hardware and software) platforms remain challenging. In particular, these platforms pose new challenges in how (i) datasets need to be pre-processed and interfaced with these platforms to achieve an optimal throughput to these devices, (ii) optimizing the runtime parameters (e.g., batch size or number of epochs for training of DL/AI tools) can dramatically alter the performance for training and inference steps, and (iii) AI/ML tools can be effectively integrated with traditional high performance computing (HPC) workloads (such as MD simulations). Furthermore, the datasets generated from protein sequence/structure databases or simulations have unique input/output characteristics which are also fundamentally different from other datasets used to benchmark ML/DL/AI tools.

Inspired by the ML community's efforts in developing benchmarks [22] for assessing the runtime performance of ML algorithms, we have created a suite of benchmarks for structural bioinformatics applications. These include both traditional ML/DL tools for: (i) protein structure determination, (ii) analyzing long timescale MD trajectories, (iii) guiding MD simulations through inference from ML/DL approaches, and (iv) modeling protein-ligand/small-molecule interactions. These applications represent an important subset of problems that are of broad interest to the entire community. The rest of the paper is organized as follows: in Section 2, we provide a brief description of each of the applications and the relevant metrics that we have used to evaluate them. In Section 3, we describe a set of relevant metrics and desired performance measures for these diverse benchmarks. Finally, in Section 4, we conclude with an overview of how these applications can be standardized in the context of emerging compute architectures.

2 BENCHMARKS

We describe a set of representative benchmarks for structural bioinformatics applications, where ML and scalable statistical inference approaches have played a vital role in enabling the scientific goals including protein structure determination, analysis of large volumes of MD simulation datasets and guiding adaptive sampling strategies to improve the sampling limitations of MD.

We distinguish benchmark application suite from benchmark metric and value. The former is comprised of a workload and associated data set. The benchmark metric is the property that is being investigated. For ML workloads in structural bioinformatics applications, metrics could be either traditional measures of system performance such as time-to-completion or peak performance, as well as scientific measures. In this paper, we will be concerned with computational as well as scientific benchmark metrics, while using the same benchmark application suite.

We first outline the design considerations for our benchmark application suite, and provide a brief description for each of the

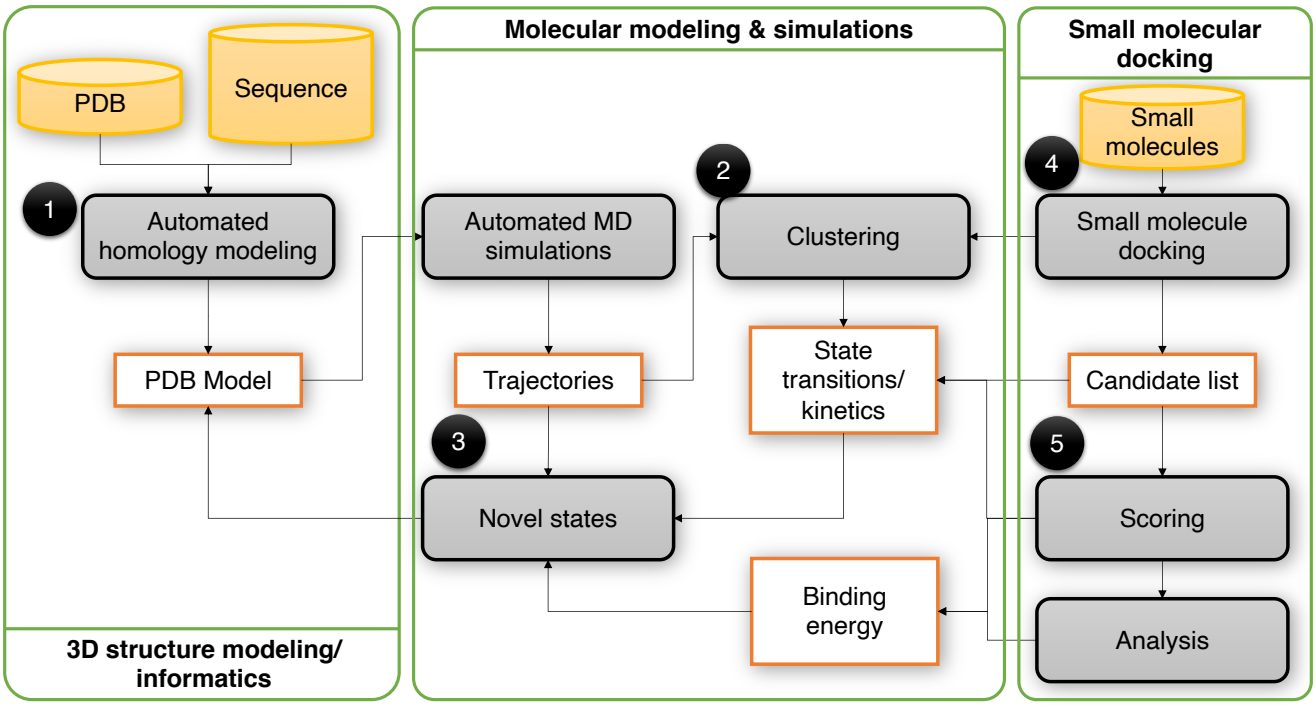

Figure 1: A reduced representation of common structural bioinformatics applications that are organized into a workflow facilitating the ultimate goal of discovering small molecules / drugs that may bind to a target bio-molecule such as proteins. The workflow components where ML approaches play a vital/enabling role are highlighted with numbers, along with representative benchmarks that were selected for this paper.

benchmarks workloads. Next, we outline the various performance metrics that we believe are useful in the context of emerging hardware and software frameworks.

2.1 Benchmarking Goals

We outline the design considerations for an initial and representative subset of applications and metrics. The overall goals of the benchmark applications selected include:

(1) Select a set of representative ML workloads that are relevant for structural bioinformatics applications. These workloads must be easily accessible and implemented.
(2) Evaluate these representative workloads using data-sets that are both representative and insightful for real structural bioinformatics scenarios, e.g., data sizes, trajectory size, length of protein sequence.
(3) Choose workloads and metrics that are insightful when evaluating the integration with traditional HPC workloads such as MD simulations.
(4) Evaluate multiple benchmark metrics, e.g., throughput (both for training and inference workloads), latency, time-to-completion, etc. The multiple metrics represent the diverse considerations and the multi-dimensionality of performance.
(5) Benchmarks workloads and metrics that can utilize leading hardware and software capabilities.

2.2 Workload Description

Figure 1 provides an overview of common structural bioinformatics applications organized into a common workflow where the ML/DL/AI approaches are highlighted: (1) homology modeling for protein structure determination; (2) clustering of conformations generated by MD (or other simulation techniques) to model state transitions and (3) identify novel states; (4) using novel states to perform small molecule docking studies where by a (5) ranked list of small molecules can be identified, and scored such that these can be further simulated using MD simulations. We have chosen a set of four representative benchmarks workloads, each of which captures a use-case for structural bioinformatics applications. We provide a brief description of these benchmarks applications below and its baseline implementation.

Recurrent geometric networks (RGN) for protein structure prediction. The overall goal can be summarized as follows: given a protein's primary sequence, determine its most likely 3D folded structure. Long standing scientific competitions such as the Critical Assessment of Protein Structures (CASP) [25] have encouraged the development of a variety of protein structure prediction methods. More recently, advances in ML techniques, specifically deep learning methods, have accelerated the development of effective protein structure prediction methods. A full assessment of the state-of-the-art techniques is out of scope for this paper; however, interested readers are referred to [13].

We chose the recurrent geometric network (RGN) [2] as a benchmark for protein structure prediction problem. The network is based on a recurrent model that uses two-layer Long Short-Term Memory (LSTMs) sequence. It takes as input amino acid sequences, where each residue is described with a one-hot encoding as well as a position-specific scoring matrices that demonstrates its mutation propensity. The first recurrent layer generates the protein backbone dihedrals for each residue both upstream and downstream from N- and C- termini. Subsequent geometric layer outputs the complete 3D structure of input sequence, based on dihedrals from previous layer. Network parameters are optimized by minimizing a distance-based root mean square deviation (dRMSD), which is Root Mean Square Deviation of pairwise distances of all atoms, between predicted and experimental structures. RGN network can achieve the state-of-art accuracy comparing to conventional sequence-to-structure maps, through physical models and fragment assembly, and it is particularly advantageous handling proteins with mutations and indels.

One of the known challenges of the RGN is that the training times can be quite long, and grows linearly with the length of the protein chain that is being modeled. Even though it achieves better accuracy when compared with existing approaches, RGN can potentially take even a week to train on modestly sized proteins on a single GPU. Conventional approaches that are based on using template based modeling strategies can take several hours to days [10].

Analyzing long timescale molecular simulations. Deep learning methods have also proven useful in extracting biophysically meaningful low dimensional representations from long time-scale MD simulations. We (and other groups) have devised unsupervised machine learning approaches to cluster MD simulation trajectories to identify conformational states [3, 9, 18, 23]. These methods utilize a variational autoencoder (VAE) as a core unit; the major differences being in how the MD data is pre-processed, or how the input data is filtered (either via convolutional layers, or LSTM layers, or Bayesian approaches).

Here we describe one of the VAE approaches that utilize convolutional filters to process the MD simulation data. The hourglass-shaped autoencoder can be trained with MD simulation trajectories in unsupervised fashion, and generates a low-dimensional representation of conformers in latent space [3]. It comprises an encoder that compresses each input into latent space and a decoder that reconstructs latent representation back to original data. Each protein conformer in molecular trajectory is described as contact map between backbone C_α atoms. The contact maps are treated as 2D images and convolutional layers are adopted to capture local patterns that correspond to protein secondary and tertiary structural features. Instead of feedforward network, a variational layer is employed at bottleneck of autoencoder that proteins conformers are encoded as a normal distribution in latent space. The overall loss consists of a reconstruction loss between original and reconstructed contact maps, and Kullback-Leibler (KL) divergence evaluating the latent normal embedding against a normal distribution with mean 0 and standard deviation 1. We have shown that this network can successfully separate different states of conformers into clusters in the latent space for further analysis, such as Markov State models.

While CVAE (and other variations) tend to work well with smaller proteins, with increasing protein sequence lengths, these networks can take a significant amount of time to train. Proteins with over 1,000 C^α atoms would entail the need to use distributed training methods, since the data will not fit into a GPU.

In addition to these techniques, several traditional ML approaches are also used to effectively reduce the dimensionality of MD datasets; however for the purposes of this paper, we focus narrowly on the use of DL approaches, since they impose substantially larger compute requirements on GPUs (and hence, with emerging ML hardware platforms).

ML-driven adaptive sampling. MD simulations can be easily trapped within local minima and therefore can become less productive in sampling new molecular states [31]. Traditionally, a user would intervene and identify if a simulation needs to stop once it reaches equilibrium or a stale state. To overcome the sampling limitations of MD, several approaches have been proposed. On the one extreme, there are approaches that have built specialized hardware to accelerate the integration step of a MD simulation. On the other hand, ensemble simulations as well as adaptive sampling strategies provide practical means to initiate a large number of simulations with different starting points and use statistical approaches to integrate these simulation data to study a variety of complex biological phenomena. However, with ensemble and adaptive sampling approaches, it is impractical to monitor which simulations must be continued versus which simulations need to be culled.

Various adaptive MD simulations have been proposed. Common to each approach is the start of an ensemble of MD runs, which is then managed iteratively based on some statistical criteria. We posited that the variational autoencoder (described above) could be used to interface with an ensemble of MD runs and make decisions to continue or terminate a simulation based on whether the run sampled different conformer states, or spawn new simulations from a less sampled configuration with available hardware (GPUs).

Our implementation, called DeepDriveMD [14], uses OpenMM [7] as the preferred MD software, which offers better performance while running on a GPU platform. Its Python API enables us to implement MD simulations with reporter that generates protein contact maps along with the trajectories. During the first phase, data generated from the MD is used to train the VAE using convolutional filters on the contact maps. The learned encoder is then used to embed the generated conformers. This embedding is then analyzed using the Density-based spatial clustering of applications with noise (DBSCAN) algorithm to identify outliers within low-density regions of the latent space. These outliers are then used to spawn new MD runs (while simultaneously culling stale runs). This enables us to sample the conformational space spanned by the protein more comprehensively than other techniques.

Reinforcement learning-driven small-molecule docking. Docking small molecules to a protein's binding site is often one of the first steps for virtual screening [16]. Although many open-source and commercial packages exist for docking, AI approaches can be equally powerful (and computationally more efficient) for docking studies [15]. Utilizing advances in control from reinforcement learning (RL), we trained an agent to drive the docking of a rigid ligand into a flexible protein pocket. The RL agent treats the ligand as a rigid body to which it can move through affine transformations along the protein. This procedure bypasses sampling on a grid as

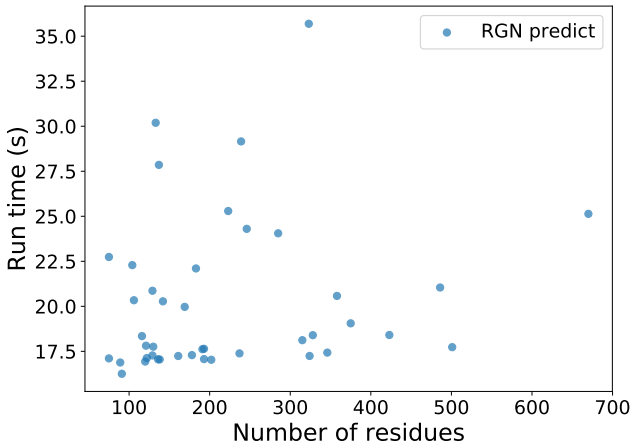

Figure 2: Runtime of RGN of CASP12 proteins. Each data point represents a protein sequence with number of residues plotted in x-axis and prediction time in y-axis.

the agent is trained to perform the optimize the pose against the standard OpenEye Fred docking function [19]. The challenge of this approach is taht there is a need to train the agent based on the protein target, which can still take considerable time on single GPU systems.

3 COMPUTATIONAL AND SCIENTIFIC METRICS

In this section, we outline the performance considerations and provide a framework of metrics that may be appropriate for structural bioinformatics applications. We have considered two classes of applications: (1) use of ML/DL/AI approaches to analyze existing, potentially large datasets (such as protein sequences, or MD trajectories), and (2) use of ML/DL/AI approaches to direct adaptive sampling MD simulation campaigns. While many of the metrics within ML-perf are widely applicable here (for approaches that are purely dependent on ML), there is a need to consider additional metrics, when ML approaches are coupled to simulations.

3.1 Initial runtimes of benchmarks

We predicted 42 (out of 90) sequences from the Critical Assessment of protein Structure Prediction (CASP) challenge using RGN, implemented with tensorflow 1.14.0 [2]. We ran the RGN using Python 2.7. Each protein sequence is preprocessed with JackHMMer again CASP12 sequence database to obtain its position-specific scoring matrix, which contains the co-evolution data for each amino acid. The encoded amino acid labels and PSSM record is then converted into TFRecord format for RGN input. The prediction is then run on a single Tesla V100 GPU with 16 GB RAM from NVIDIA DGX-1 machine.

Note that this benchmark tests only the prediction time it takes for modeling a sequence (rather than the training time of the RGN itself). As shown in Fig. 2, the prediction times do not show a clear trend; however, we note that even for moderately sized proteins, the prediction time can be quite large. In Fig. 3 we present initial

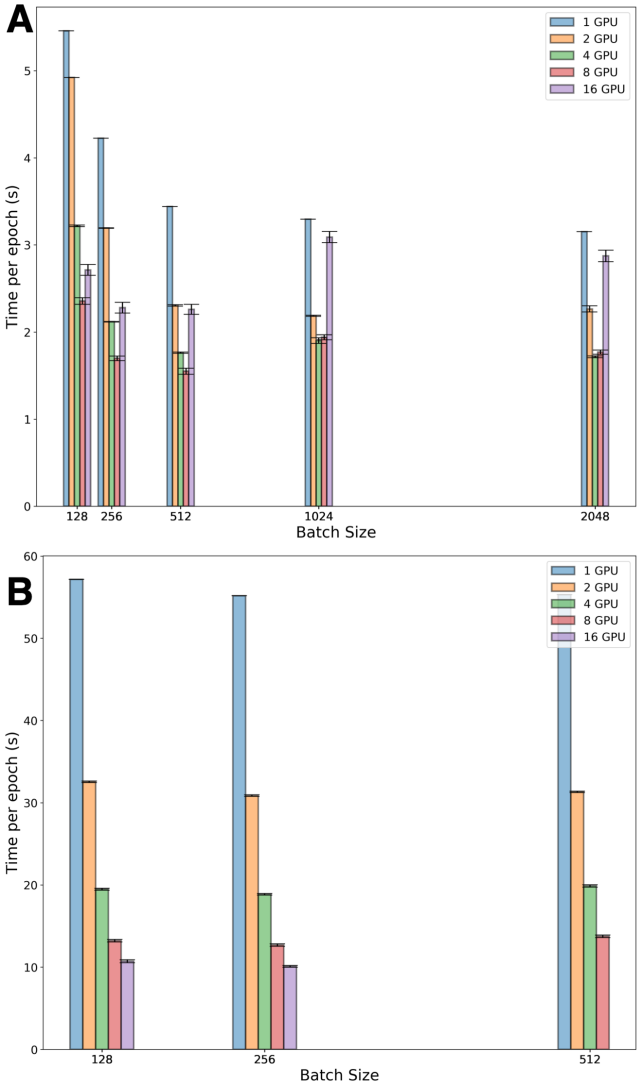

Figure 3: Runtime characteristics of CVAE using data-parallel (distributed) training implemented with horovod for two protein simulations. (A): shows the performance of Horovod in using larger batch sizes for the Fs-peptide protein system where the input data is represented as a 22 × 22 residue contact matrix. (B): shows the performance for a larger system where the input data is represented as a 106 × 106 contact matrix. In both cases, we analyze the results using a CVAE – whereas we can use batch sizes of 2048 for the smaller protein system, the batch sizes are significantly smaller for the larger protein system.

timing runs for training the CVAE on larger batch sizes (as generated from a MD simulation). This approach uses distributed deep learning implementation using the Horovod library [26]. We used Tensorflow version 2.0.0. The raw data was pre-processed, converted to TFRecord format and stored as a list of bytes (for efficient processing of the input data). The data was trained and tested using the CVAE algorithm adapted to run with Horovod for distributed

deep learning across multiple GPUs. Python 3.7.0 was used, training 10 epochs per run. The training was run using a NVIDIA DGX-2 machine, which has a total of 16 Tesla V100 GPUs connected on to two Intel processors.

The runs illustrate some challenges in effectively scaling the CVAE for larger protein systems. First, we observe that increasing the number of GPUs reduces the training time, albeit in a non-linear manner. However, for the Fs-peptide system, we observe that the scaling does not hold – indicating perhaps time spent in communication overheads, rather than actual computation. This is however different for the larger system size, where increasing the number of GPUs also improves the training time per epoch. While convolutional filters run faster (and are optimized for GPUs), distributed runs present additional challenges in scaling effectively on the hardware. As we have also indicated in previous work, the ability to keep the GPU effectively occupied during distributed training runs can be quite challenging [34].

3.2 Computational Metrics

Apart from evaluating the usual metrics of accuracy and clustering effectiveness for each of ML/DL/AI approaches, as well as measures determined in community benchmarks [22], there is a need to assess two other important metrics that are specific to our applications: (1) the latency in inference mode for the given ML/DL/AI approaches, and (2) data transfer efficiency – a measure that provides a bound on how long it will take to transfer the data between compute nodes that run the ML/DL/AI approaches and the simulations (and vice versa).

For (1), the latency in the inference mode is mainly determined by the fact that simulations generate $O(1,000)$ conformations every couple of minutes or so (depending on the system size and simulation times). Thus, the Ml/DL/AI inferences have to be tuned to the data generation rates from the simulations. This is again dependent on how simulation datasets are represented and routed for analysis (for e.g., data compression or having data generators can significantly accelerate how the data is passed between compute nodes). Further, with event identification tasks, typically, both the training and inference approaches need to be run *online* or *in situ*, which entails additional constraints on the use of the ML/DL/AI approaches.

For (2), given that we are making implicit use of ensemble MD simulations, there is a definite skew in terms of the data generated (from simulations) and how it is transferred between compute nodes. This latency also determines the effective performance (see below) of the sampling approaches; if the data cannot be transferred efficiently, then even if the ML/DL/AI inferences are fast enough, the limiting step would be that of the data transfer.

In addition to these metrics, it is important to understand how precision (of floating point calculations) can affect the overall performance of these workflows. Since ML/DL/AI approaches generally tend to achieve higher performance (mainly speed in inference throughput), it may be tempting to use low precision for inference tasks. However, with low precision representations, this can significantly affect the MD simulations downstream (due to energy drifts or other artifacts with low precision runs). Thus, such interfaces also need to be carefully evaluated.

3.3 Scientific Metrics for ML-driven Adaptive Sampling

As a representative example of a benchmark metric that captures scientific performance, we consider the advanced sampling problem in biophysics. The fundamental challenge that sampling algorithms address, is how to effectively traverse a very large and high-dimensional phase space without exhaustively visiting every part of it. The computational biophysics community has invested significant effort and energy into developing sophisticated sampling algorithms that sample phase space in ways that can be characterized as being *better, faster or greater*.

Although better, faster and greater are contextual, canonical descriptions are provided: (i) *Better*: Sample parts of phase space that have greater contribution to the partition function or are relatively more important to compute a physical property more accurately; (ii) *Faster*: Sample the relevant part of phase space or a certain fixed portion of phase space in a shorter period of time (which here is amount of computational capacity), and (iii) *Greater*: Span a larger part of phase in given period of time (or using a given computational capacity).

To determine the effectiveness of ML-driven sampling approaches, they need to be benchmarked against "classical" or "vanilla" sampling algorithms. Further, the amount of available computational resources is concomitant to and implicitly associated with the notion of sampling effectiveness. This introduces the concept of "effective sampling", which is crudely approximated as the ratio of ML-driven sampling to classical sampling for a given computational resource. Different physical measures, viz., RMSD, Schlitter Entropy, Poincare recurrence time can be computed and the type of sampling effectiveness can be better || faster || greater.

4 CONCLUSIONS

In this position paper, we have presented an initial set of benchmarks that may be useful to evaluate the coupling of ML/DL/AI tasks with physics-based simulations. The proposed benchmarks are inspired by the recent developments in benchmarking ML approaches (e.g.,ML-perf), as well as the need to develop best practices for molecular simulation toolkits (for both biological and materials science applications). As ML tools continue to interface with MD simulation toolkits, closer integration and co-design of software that can interface seamlessly with emerging heterogenous compute platforms is a necessity. In particular, the heterogeneous nature of the individual workflow components means that having meaningful benchmarks could serve the broader community in developing tools that can be sustainable into the future.

The proposed benchmarks applications and metrics are representative but not complete. We believe that a community wide effort is essential in further developing these benchmarks. We also hope that the list of applications described here can serve the community to contribute other benchmarks applications and datasets. The benchmarks presented here are being compiled into a Github repository at: https://github.com/DeepLearn-Benchmark; we expect to release this to the community as soon as common interfaces and functionality are defined.

Acknowledgements

This material is based upon work supported by the U.S. Department of Energy, Office of Science, under contract number DE-AC02-06CH11357. This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility. We thank Prof. Emad Tajkhorshid from the NIH Center for Macromolecular Modeling and Bioinformatics for providing access to the NVIDIA DGX-2 machine, which was supported by NIH P41-GM104601 grant. Anda Trifan acknowledges support from the United States Department of Energy through the Computational Sciences Graduate Fellowship (DOE CSGF) under grant number: DE-SC0019323. We thank Hyungro Lee, Matteo Turilli and other members of the RADICAL Laboratory for support of DeepDriveMD.

REFERENCES

[1] Stewart A. Adcock and J. Andrew McCammon. 2006. Molecular dynamics: Survey of methods for simulating the activity of proteins. *Chem Rev* 106, 5 (2006), 1589–1615.

[2] Mohammed AlQuraishi. 2019. End-to-End Differentiable Learning of Protein Structure. *Cell Systems* 8, 4 (2020/01/16 2019), 292–301.e3. https://doi.org/10.1016/j.cels.2019.03.006

[3] Debsindhu Bhowmik, Shang Gao, Michael T. Young, and Arvind Ramanathan. 2018. Deep clustering of protein folding simulations. *BMC Bioinformatics* 19, 18 (21 Dec 2018), 484. https://doi.org/10.1186/s12859-018-2507-5

[4] Gregory R. Bowman, Kyle A. Beauchamp, George Boxer, and Vijay S. Pande. 2009. Progress and challenges in the automated construction of Markov state models for full protein systems. *J Chem Phys* 131, 12, Article 124101 (2009), 11 pages. https://doi.org/10.1063/1.3216567

[5] D. A. Case, T. A. Darden, T. E. III Cheatham, C. L. Simmerling, J. Wang, R. E. Duke, R. Luo, K. M. Merz, B. Wang, D. A. Pearlman, M. Crowley, S. Brozell, V. Tsui, H. Gohlke, J. Mongan, V. Hornak, G. Cui, P. Beroza, C. Schafmeister, J. W. Caldwell, W. S. Ross, and P. A. Kollman. 2012. AMBER 12. *University of California, San Francisco* (2012).

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

[7] 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. 2017. OpenMM 7: Rapid development of high performance algorithms for molecular dynamics. *PLOS Computational Biology* 13, 7 (07 2017), 1–17. https://doi.org/10.1371/journal.pcbi.1005659

[8] Katherine A. Henzler-Wildman, Ming Lei, Vu Thai, S. Jordan Kerns, Martin Karplus, and Dorothee Kern. 2007. A hierarchy of timescales in protein dynamics is linked to enzyme catalysis. *Nature* 450, 7171 (12 2007), 913–916.

[9] C. X. Hernández, H. K. Wayment-Steele, M. M. Sultan, B. E. Husic, and V. S. Pande. 2017. Variational Encoding of Complex Dynamics. *ArXiv e-prints* (Nov. 2017). arXiv:stat.ML/1711.08576

[10] Shaun M. Kandathil, Joe G. Greener, and David T. Jones. 2019. Recent developments in deep learning applied to protein structure prediction. *Proteins: Structure, Function, and Bioinformatics* 87, 12 (2019), 1179–1189. https://doi.org/10.1002/prot.25824 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/prot.25824

[11] M. Karplus and C.B. Post. 1996. Simulations of lysozyme: internal motions and the reaction mechanism. *EXS* 75 (1996), 111–141.

[12] John L Klepeis, Kresten Lindorff-Larsen, Ron O Dror, and David E Shaw. 2009. Long-timescale molecular dynamics simulations of protein structure and function. *Current Opinion in Structural Biology* 19, 2 (2009), 120 – 127. https://doi.org/10.1016/j.sbi.2009.03.004 Theory and simulation / Macromolecular assemblages.

[13] Brian Kuhlman and Philip Bradley. 2019. Advances in protein structure prediction and design. *Nature Reviews Molecular Cell Biology* 20, 11 (2019), 681–697. https://doi.org/10.1038/s41580-019-0163-x

[14] Hyungro Lee, Matteo Turilli, Shantenu Jha, Debsindhu Bhowmik, Heng Ma, and Arvind Ramanathan. 2019. DeepDriveMD: Deep-Learning Driven Adaptive Molecular Simulations for Protein Folding. In *2019 IEEE/ACM Third Workshop on Deep Learning on Supercomputers (DLS)*. IEEE, 12–19.

[15] Jin Li, Ailing Fu, and Le Zhang. 2019. An Overview of Scoring Functions Used for Protein–Ligand Interactions in Molecular Docking. *Interdisciplinary Sciences: Computational Life Sciences* 11, 2 (2019), 320–328. https://doi.org/10.1007/s12539-019-00327-w

[16] Paul D Lyne. 2002. Structure-based virtual screening: an overview. *Drug Discovery Today* 7, 20 (2002), 1047 – 1055. https://doi.org/10.1016/S1359-6446(02)02483-2

[17] Heng Ma, Debsindhu Bhowmik, Hyungro Lee, Matteo Turilli, Michael T Young, Shantenu Jha, and Arvind Ramanathan. 2019. Deep generative model driven protein folding simulation. *arXiv preprint arXiv:1908.00496* (2019).

[18] Andreas Mardt, Luca Pasquali, Hao Wu, and Frank Noé. 2018. VAMPnets for deep learning of molecular kinetics. *Nature Communications* 9, 1 (2018), 5. https://doi.org/10.1038/s41467-017-02388-1

[19] Mark McGann. 2011. FRED Pose Prediction and Virtual Screening Accuracy. *Journal of Chemical Information and Modeling* 51, 3 (03 2011), 578–596. https://doi.org/10.1021/ci100436p

[20] A. Ramanathan, A. Savol, V. Burger, C. S. Chennubhotla, and P. K. Agarwal. 2013. Protein Conformational Populations and Functionally Relevant Substates. *Acc. Chem. Res.* (Aug 2013).

[21] Arvind Ramanathan, Andrej Savol, Virginia Burger, Shannon Quinn, Pratul K Agarwal, and Chakra Chennubhotla. 2012. Statistical inference for big data problems in molecular biophysics. In *Neural Information Processing Systems: Workshop on Big Learning*.

[22] Vijay Janapa Reddi, Christine Cheng, David Kanter, Peter Mattson, Guenther Schmuelling, Carole-Jean Wu, Brian Anderson, Maximilien Breughe, Mark Charlebois, William Chou, et al. 2019. Mlperf inference benchmark. *arXiv preprint arXiv:1911.02549* (2019).

[23] João Marcelo Lamim Ribeiro, Pablo Bravo, Yihang Wang, and Pratyush Tiwary. 2018. Reweighted autoencoded variational Bayes for enhanced sampling (RAVE). *The Journal of Chemical Physics* 149, 7 (2018), 072301. https://doi.org/10.1063/1.5025487 arXiv:https://doi.org/10.1063/1.5025487

[24] Martin K. Scherer, Benjamin Trendelkamp-Schroer, Fabian Paul, Guillermo Pérez-Hernández, Moritz Hoffmann, Nuria Plattner, Christoph Wehmeyer, Jan-Hendrik Prinz, and Frank Noé. 2015. PyEMMA 2: A Software Package for Estimation, Validation, and Analysis of Markov Models. *Journal of Chemical Theory and Computation* 11 (Oct. 2015), 5525–5542. https://doi.org/10.1021/acs.jctc.5b00743

[25] Andrew W. Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander W. R. Nelson, Alex Bridgland, Hugo Penedones, Stig Petersen, Karen Simonyan, Steve Crossan, Pushmeet Kohli, David T. Jones, David Silver, Koray Kavukcuoglu, and Demis Hassabis. 2019. Protein structure prediction using multiple deep neural networks in the 13th Critical Assessment of Protein Structure Prediction (CASP13). *Proteins: Structure, Function, and Bioinformatics* 87, 12 (2019), 1141–1148. https://doi.org/10.1002/prot.25834 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/prot.25834

[26] Alexander Sergeev and Mike Del Balso. 2018. Horovod: fast and easy distributed deep learning in TensorFlow. arXiv:cs.LG/1802.05799

[27] Zahra Shamsi, Kevin J. Cheng, and Diwakar Shukla. 2018. Reinforcement Learning Based Adaptive Sampling: REAPing Rewards by Exploring Protein Conformational Landscapes. *The Journal of Physical Chemistry B* 122, 35 (09 2018), 8386–8395. https://doi.org/10.1021/acs.jpcb.8b06521

[28] David E. Shaw, Martin M. Deneroff, Ron O. Dror, Jeffrey S. Kuskin, Richard H. Larson, John K. Salmon, Cliff Young, Brannon Batson, Kevin J. Bowers, Jack C. Chao, Michael P. Eastwood, Joseph Gagliardo, J. P. Grossman, C. Richard Ho, Douglas J. Ierardi, István Kolossváry, John L. Klepeis, Timothy Layman, Christine McLeavey, Mark A. Moraes, Rolf Mueller, Edward C. Priest, Yibing Shan, Jochen Spengler, Michael Theobald, Brian Towles, and Stanley C. Wang. 2008. Anton, a Special-purpose Machine for Molecular Dynamics Simulation. *Commun. ACM* 51, 7 (July 2008), 91–97. https://doi.org/10.1145/1364782.1364802

[29] Marcos Sotomayor and Klaus Schulten. 2007. Single-Molecule Experiments in Vitro and in Silico. *Science* 316, 5828 (2007), 1144–1148. https://doi.org/10.1126/science.1137591 arXiv:https://science.sciencemag.org/content/316/5828/1144.full.pdf

[30] H. Stamati, C. Clementi, and L. Kavraki. 2010. Application of nonlinear dimensionality reduction to characterize the conformational landscape of small peptides. *Proteins: Struct Funct Bioinform* 78 (2010), 223–235.

[31] Yihang Wang, Joao Marcelo Lamim Ribeiro, and Pratyush Tiwary. 2019. Machine learning approaches for analyzing and enhancing molecular dynamics simulations. arXiv:physics.comp-ph/1909.11748

[32] Yihang Wang, João Marcelo Lamim Ribeiro, and Pratyush Tiwary. 2019. Past–future information bottleneck for sampling molecular reaction coordinate simultaneously with thermodynamics and kinetics. *Nature Communications* 10, 1 (2019), 3573. https://doi.org/10.1038/s41467-019-11405-4

[33] Jeffrey K. Weber and Vijay S. Pande. 2011. Characterization and Rapid Sampling of Protein Folding Markov State Model Topologies. *J Chem Theory Computat* 7, 10 (2011), 3405–3411. PMID: 22140370.

[34] S. Yoginath, M. Alam, A. Ramanathan, D. Bhowmik, N. Laanait, and K. S. Perumalla. 2019. Towards Native Execution of Deep Learning on a Leadership-Class HPC System. In *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. 941–950. https://doi.org/10.1109/IPDPSW.2019.00160