AccelMD: A Self-adaptive AI-enabled Framework for Accelerating Molecular Dynamics Simulations

Kazi Fahim Ahmad Nasif*, Bobin Deng*, Lingtao Chen*, Yixin Xie*, Shaolei Teng‡, Jiang Li§, Liang Zhao*, Syed Md Shamsul Alam†, Nobel Dhar*, Kun Suo*, Dan Chia-Tien Lo*

* College of Computing and Software Engineering, Kennesaw State University

knasif@students.kennesaw.edu, bdeng2@kennesaw.edu, lchen25@students.kennesaw.edu, {yxie11, lzhao10}@kennesaw.edu, ndhar@students.kennesaw.edu, {ksuo, dlo2}@kennesaw.edu

† School of Informatics, Computing, and Cyber Systems, Northern Arizona University

ss5687@nau.edu

‡ Department of Biology, Howard University

shaolei.teng@Howard.edu

§ Department of Electrical Engineering and Computer Science, Howard University

jli@Howard.edu

***Abstract*—Molecular Dynamics (MD) simulation is a fundamental exploring approach for numerous scientific fields, including but not limited to drug discovery, biology, material science, and chemistry. Unfortunately, large-scale MD simulation generally requires long-period processing, even in resource-intensive supercomputers. Parallel computing is the primary methodology to accelerate MD simulation. However, parallel computing has a low theoretical improvable ceiling for MD simulation because the atom statuses of one timestep depend on its previous timestep. To unlock the full potential of MD simulation in drug discovery and biology, we propose an AccelMD framework for reducing processing time and computing costs. AccelMD is an AI-enabled framework that accelerates MD simulation while maintaining high predictive accuracy. In order to extend AccelMD's applicability and flexibility, we also developed an extra preprocessing stage, which allows the framework to adapt to various protein inputs automatically. Compared to conventional MD, the AccelMD framework achieves *47.59*X speedup on average (max: *63.78*X, min: *30.61*X). Regarding the prediction accuracy of protein structures, the average Mean Absolute Error (MAE) and TM-Score values of AccelMD are *0.0058* and *0.99980*, respectively. Our empirical experiments exihibit the robustness of AccelMD framework for the study of complex, long-timescale molecular interactions.**

***Index Terms*—Molecular Dynamics, AI, Protein Structure Prediction, Accelerating Simulation.**

I. Introduction

Molecular dynamics (MD) simulations predict how every atom in a protein or other molecular system move over time [1]. By using physics based formulas and rules MD simulations can significantly reduce the overload of physical experiments. But solving complex calculations iteratively requires a lot of computational overhead. Because simulations operate on femtoseconds (10^{-15}s) timescales, modeling biologically relevant events that occur over microseconds or longer is a grand challenge, often requiring months of processing on high-performance computers (HPCs). Each iteration is also a complicated process that includes force field calculations from particle positions, solving Newton's equations of motion, and moving data in the memory hierarchy. The situation becomes even worse when the number of atoms increases for larger molecular inputs. Therefore, exploring the long-term molecular events, such as protein folding or large conformational changes [2], via conventional MD simulation is challenging, even processing in resource-intensive High-performance Computers (HPCs). E.g., a 20 nanoseconds MD simulation (*OpenMM* [3]) for a *13,137* atoms and 1653 residues' molecular system requires 05:00 hours in our HPC that integrates with 256 AMD EPYC 7742 64-Core Processor CPU, eight NVIDIA A100-SXM4-80 GB GPU, and 2 TB RAMs. The improvement of computer system performance is relatively slow, and it is hard to meet the computational requirements for conventional MD simulation. Therefore, the computing resource bottleneck limits the continuous development of scientific domains, which typically require large-scale molecular simulations or long-term medium-size simulations.

Parallel computing offers some acceleration in molecular system study [4]; however, as MD simulations are sequential in nature, parallel computing has inherent limitations for significant acceleration. Consequently, AI-based methods are getting popular in MD simulations. However, most related approaches focus on creating Neural Network Potentials (NNPs) to predict accurate energies functions and forces fields. These methods do not directly address the challenge of accelerating the long-term protein evolution over time.

A key feature is its self-adaptive preprocessing stage, which allows AccelMD to work directly with standard Protein Data Bank (PDB) files of vastly different sizes, from under 100 to nearly 3000 residues. After training on the initial dynamically collected data, our AccelMD can rapidly generate long-timescale trajectories that capture complex events like large conformational changes. Unlike tools such as AlphaFold3 [5], which predict static structures from a sequence, AccelMD focuses exclusively on accelerating the dynamic simulation.

The main contributions of this paper are as follows:

- Designing a lightweight AI framework (AccelMD) to predict dynamic protein structures during MD simulations, significantly reducing processing time (*47.59*X speedup on average) while maintaining high prediction accuracy.
- Developing a dynamic dataset creation approach specifically tailored for training our AccelMD. The generated dataset consists of time-series atomic coordinates from MD simulations.
- Implementing a self-adaptive preprocessing stage that enables the framework to handle diverse protein structures directly from PDB files automatically.

The remaining sections of the paper are outlined as follows. Section II first provides the background of the MD simulations and related optimization approaches. Section III describes AccelMD's design and implementation details. Section IV discusses our evaluation methodology and analyzes the experimental results. Finally, we conclude in Section V.

II. Related Work

MD simulations have revolutionized the study of molecular systems by providing in-depth understanding of molecular behaviors at the atomic level by Newton's laws of motion [6]. Physical experimental methods like X-ray crystallography, Nuclear Magnetic Resonance (NMR), and Cryo-Electron Microscopy (Cryo-EM) [7]–[9] have limitations in case of difficult protein crystallization of smaller proteins, and lower resolution for specific bio-molecules. Molecular dynamics (MD) simulations predict the movement of atoms in a molecular system applying the physics based rules of atomic interactions. MD simulations cannot 100% replace the traditional approach but they can significantly accelerate molecular system exploration.

MD simulations process have four fundamental steps. The first step is to prepare the system by providing the initial configuration. The next steps are responsible to minimize the energy of the system followed by bringing the system equilibrium. The final step delivers the trajectory from one of the candidate equilibrium ensemble. Analysis of the resulting trajectory provides insights into molecular behavior over time. State-of-the-art software packages for MD simulations include *GROMACS*, *NAMD*, *AMBER*, *LAMMPS* and a python based tool called *OpenMM* [3], which is highly customizable and optimized for GPU computing.

Parallel computing is a common method used to accelerate large molecular systems. However, this approach can not bring fruitful solution in MD simulations as current state of atomic structure depends on the previous state.

The most significant advent of AI in the MD simulation is the introduction of Neural Network Potentials (NNPs) [10]–[12] that can predict machine-learned forcefields using Schrödinger equation. While NNPs enable a swift, accurate, and generalizable solution for researchers developing the next generation of small-molecule drugs, they do not inherently solve the long-timescale sampling problem. The second major area is static structure prediction. AlphaFold [5] has achieved unprecedented accuracy in predicting a protein's most stable 3D conformation from amino acid sequences. However, its focus remains on predicting a single static state, not the dynamic protein interactions over time.

Our work represents a third, distinct direction: using AI to directly accelerate molecular motion simulation. Instead of predicting forces or final structures, AccelMD learns the underlying patterns of a protein's dynamics to forecast its entire structural trajectory, thereby skipping the vast majority of brute-force calculations.

III. AccelMD Framework Design and Implementation

This section introduces the design and implementation details of the AccelMD framework. Figure 1 describes the process of speeding up MD simulations using our AccelMD. This AccelMD framework is self-adaptable to random/unknown inputs (PDB files) and consists of three stages: 1) Dataset Creation 2) AccelMD Training, and 3) AccelMD Inference.

A. Dynamic Dataset Creation

A key aspect of AccelMD is its in situ training approach, which begins by generating a dataset that captures the input protein's unique dynamic fingerprint. This is achieved by running a tiny, *20 nanoseconds (ns)* conventional MD simulation. From this trajectory, we format the data for training using a sliding window mechanism, where a sequence of past structures is used to teach the model how to predict the immediate next frame. The remaining long-term dynamics are then predicted by the trained AccelMD model, eliminating the need for further brute-force simulation.

As illustrated in Figure 1, The dataset creation process involves multiple sequential substeps. The OpenMM library [13] is utilized for initial MD simulation and dataset preparation. Our AccelMD approach is not limited to OpenMM; it can seamlessly extend to the majority of regular MD simulation tools. Protein Data Bank (PDB) files [14] are the inputs of MD simulations. Unfortunately, PDB files frequently lack atoms, residues, and bonds. PDBFixer tool [15] is employed to identify and recover any missing residues, replace nonstandard residues with standard ones, and remove heteroatoms while retaining water molecules. Missing atoms within residues are added, and hydrogens are introduced at a specific pH to ensure proper protonation states. The fixed PDB files were then used to run an initial *20 ns* MD simulation using a standard set of parameters (e.g., a Langevin integrator at *300 K* with a *2 fs* timestep), with coordinates saved every *10,000* steps for analysis.

As the energy plots in Figure 2a demonstrate that the molecular systems typically reach to equillibrium state within the first 5-10 ns, an initial simulation of 20ns is enough for making the training dataset. Once the system's energy has stabilized, the simulation can effectively sample the protein's characteristic conformational states and primary modes of motion. Our observations across a diverse set of proteins, ranging from small to large, verified that the 20 ns period is generally sufficient for the AI model to learn the essential

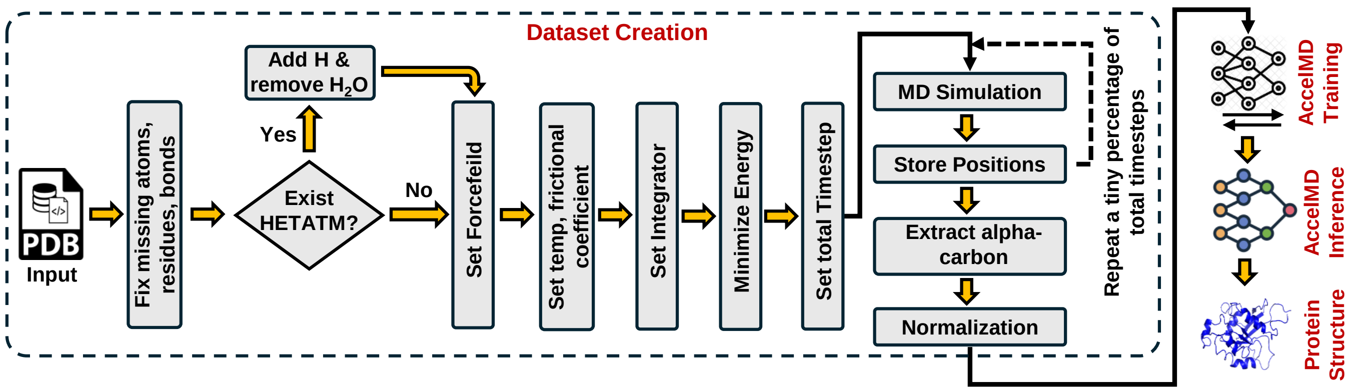

Fig. 1: The Workflow of Utilizing AccelMD Framework for Accelerating MD Simulations

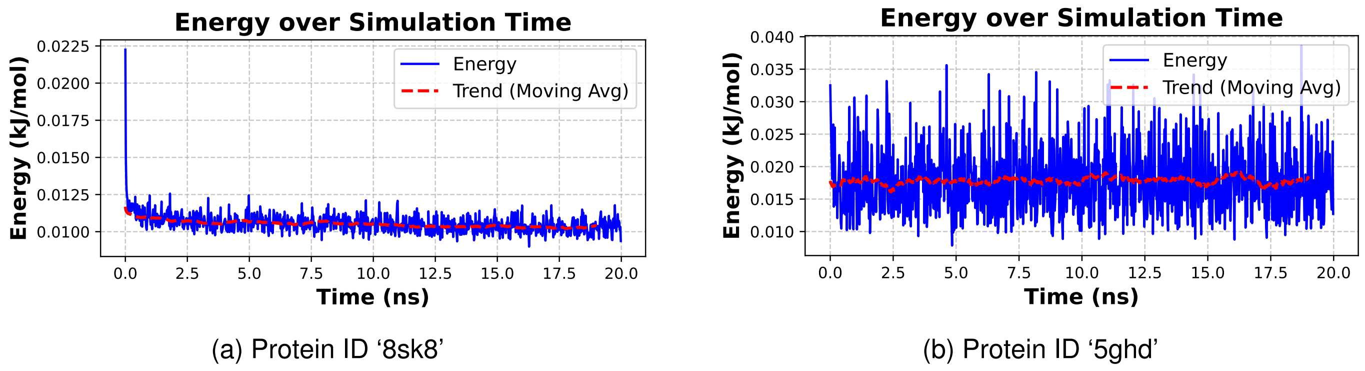

Fig. 2: Energy simulation over time with trend for two protein structures.

patterns of the target molecular movements. This learned dynamic pattern is the foundation for precise long-term protein prediction.

After every *10,000* steps of conventional MD simulation, we save all atomic coordinates of this protein structure along with the associated metadata such as *atom_names*, *residue_names*, and *residue_numbers*. Each time step is *0.002* picoseconds, total *1000* data points are collected in the Angstrom unit over this *20* ns simulation time. Each residue contains one alpha-carbon. As only alpha-carbons determine the structure of the protein, so we extracted the alpha-carbons' coordinate values from each residue for the dataset. The whole dataset is organized in a 3D numpy array (dimensions: *(number_of_datapoints, number_of_residues, 3)*). In other words, our dataset contains alpha-carbons' coordinates of all residues for every timestep. For example, for a PDB file containing *1653* residues the dataset will look like '*Timestep 1, residue 1, (x,y,z)*', '*Timestep 1, residue 2, (x,y,z)*',..., '*Timestep 1000, residue 1652, (x,y,z)*', and '*Timestep 1000, residue 1653, (x,y,z)*'.

B. AccelMD Model Architecture

The AccelMD architecture is built upon a Convolutional Neural Network (CNN) enhanced with a self-attention mechanism. We chose this design over prevalent sequence-modeling architectures, such as RNNs, LSTMs, GRUs and state-of-the-art Transformer-based models designed for time-series forecasting because MD trajectory data is inherently multi-dimensional, not just multivariate. Each simulation step look like a moving video of sequential 3D frames. We considered each frame like an image and converted the outputs in a grid like format. Then applied CNN model to capture the pattern of the simulation followed by a self-attention layer to learn the long-range interactions between distant atoms, overcoming the limited receptive field of standard convolutions and enhancing predictive accuracy.

Figure 3 shows the overview of AccelMD architecture, which consists of two convolutional layers, a self-attention layer, two max-pooling layers, two batch normalization layers, two fully connected dense layers, and a reshape layer. The AccelMD architecture begins with self-adaptive preprocessing step that organizes the atomic coordinate data into a square grid format ($g \times g \times 3$; '$g \times g$' is the total coordinate points, and '*3*' indicates each coordinate point contains *3* values: *x*, *y*, and *z*) based on the total number of residues in the protein. This preprocessing step makes the proteins of various sizes compatible for the input for the Conv2D layers. Section III-D will discuss more details about this preprocessing step.

The core AccelMD architecture comprises multiple layers for effective feature extractions and predictions. The output of the preprocessing stage is forwarded into the *Conv2D* layer and then a *MaxPooling2D* layer downsamples the data. A *BatchNormalization* layer is then applied to stabilize and accelerate training. The first *Conv2D* layer applies 32 3×3

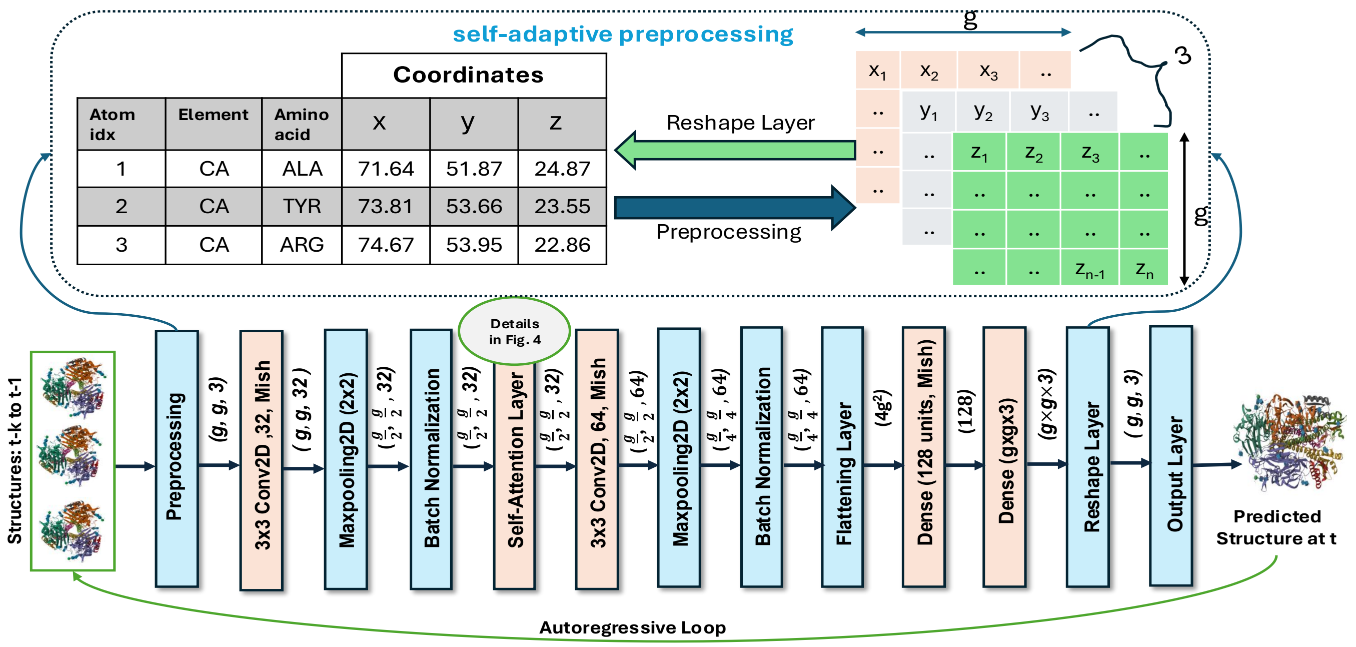

Fig. 3: AccelMD model consists of two convolutional layers, a self-attention layer, two max-pooling layers, two batch normalization layers, two fully connected dense layers, and a reshape layer. The grid size g represents the dimension of the protein input.

filters with *Mish* activation, resulting in a (g, g, 32) output shape. Each output value of the first *Conv2D* layer is calculated as:

$$Output = Mish(Input * Kernel + Bias)$$
$$Mish(x) = x \cdot \tanh(\ln(1 + e^x)) \tag{1}$$

The first MaxPooling2D layer downsamples the $(g, g, 32)$ output to $(\frac{g}{2}, \frac{g}{2}, 32)$ by taking the max in each 2×2 window. The BatchNormalization layer normalizes the $(\frac{g}{2}, \frac{g}{2}, 32)$ output to stabilize learning:

$$Output[i,j,k] = \gamma \left(\frac{Input[i,j,k] - \mu_k}{\sigma_k} \right) + \beta \tag{2}$$

To enhance the ability to capture long-range dependencies and atom interactions, we integrate a self-attention mechanism in the AccelMD. The *Self-Attention* layer, with an $(\frac{g}{2}, \frac{g}{2}, 32)$ input, calculates the layer output as below:

$$Q = Dense(X), K = Dense(X), V = Dense(X)$$
$$Scores = \frac{Q \cdot K^T}{\sqrt{d_k}}$$
$$Weights = softmax(Scores) \tag{3}$$
$$Output = Weights \cdot V$$

AccelMD employs a multi-head attention mechanism with eight heads (Figure 4) like the original Transformer [16], to capture features from different representation subspaces. The attention layer's output is then passed through a final convolutional block which contains a *Conv2D* layer with of 64 3×3 filters, followed by a *MaxPooling2D* and a *BatchNormalization* layer respectively. Finally, this block's output is flattened and passed to a fully connected dense layer with 128 neurons and a Mish activation function.

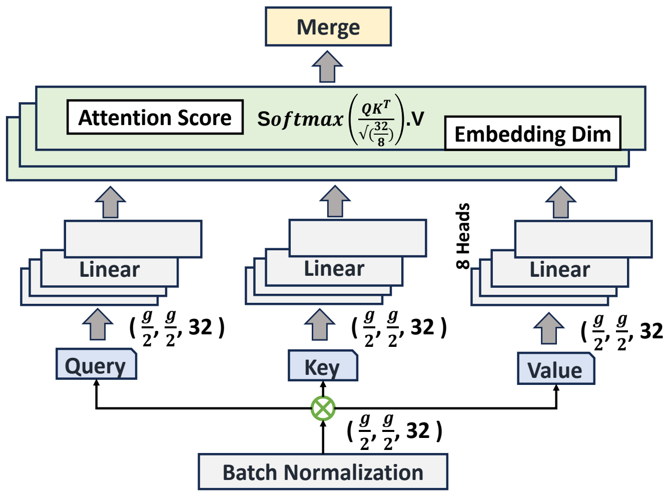

Fig. 4: Architecture of the Self-Attention Layer.

$$DenseOutput = Mish(Weights \cdot FlattenedOutput + Bias) \tag{4}$$

The final fully connected dense layer adjusts the output size to match the input of *Reshape Layer*. The *Reshape Layer* converts the 1D input array to a 3D array containing all atomic coordinates of the predicted structure. The AccelMD architecture allows dynamic reshaping inputs, making it automatically adaptable to various initial structures (PDB files). To better understand AccelMD architecture, we provide Algorithm 1 that summarizes the AccelMD computing process.

C. AccelMD Training and Inference

The basic idea of our AccelMD framework is to execute a first tiny percentage of original simulation cycles to create a dataset which is then used to train the AccelMD and predict (inference) the protein structures of future simulation cycles.

Algorithm 1 AccelMD Model

```
Require: Atomic coordinates dataset $D$ with $N$ alpha-carbon atoms and $T$ timesteps
Ensure: Predicted atomic coordinates
 1: Input: $D \in \mathbb{R}^{T\times N\times 3}$
 2: Output: Predicted coordinates $\hat{D} \in \mathbb{R}^{T\times N\times 3}$
 3: Compute the grid size $G$ as $\lceil\sqrt{N}\rceil$
 4: Reshape $D$ to $\mathbb{R}^{T\times G\times G\times 3}$ by padding with zeros
 5: Layer 1: Convolutional Layer
 6: Apply 32 $3\times 3$ filters with Mish and same padding
 7: for all $(i, j, k) \in G\times G\times 32$ do
        $$O_{ijk} = \text{Mish}\left(\sum_{m,n,c} I_{i+m,j+n,c}\cdot K_{mnck} + b_k\right)$$
 8: end for
 9: Layer 2: MaxPooling Layer
10: Apply max pooling with pool size $2\times 2$ and stride 2
11: Downsample the output to $\mathbb{R}^{\frac{G}{2}\times\frac{G}{2}\times 32}$
12: Layer 3: Batch Normalization
13: Normalize the output to stabilize learning
        $$\text{Output} = \gamma\left(\frac{\text{Input}-\mu}{\sigma}\right)+\beta$$
14: Layer 4: Self-Attention
15: Compute projections: $Q = W_Q X, K = W_K X, V = W_V X$
16: Compute attention scores $S = \frac{Q\cdot K^T}{\sqrt{d_k}}$
17: Compute attention weights $W = \text{softmax}(S)$
18: Compute the output $O = W \cdot V$
19: Layer 5: Convolutional Layer
20: Apply 64 $3\times 3$ filters with Mish and same padding
21: for all $(i, j, k) \in \frac{G}{2}\times\frac{G}{2}\times 64$ do
        $$O_{ijk} = \text{Mish}\left(\sum_{m,n,c} I_{i+m,j+n,c}\cdot K_{mnck} + b_k\right)$$
22: end for
23: Layer 6: MaxPooling Layer
24: Apply max pooling with pool size $2\times 2$ and stride 2
25: Downsample the output to $\mathbb{R}^{\frac{G}{4}\times\frac{G}{4}\times 64}$
26: Layer 7: Batch Normalization
27: Normalize the output to stabilize learning
        $$\text{Output} = \gamma\left(\frac{\text{Input}-\mu}{\sigma}\right)+\beta$$
28: Layer 8: Flatten Layer
29: Flatten the output to a 1D vector of shape $\mathbb{R}^{\frac{G^2}{16}\times 64}$
30: Layer 9: Dense Layer
31: Apply a dense layer with 128 units and Mish activation
        $$\text{Output} = \text{Mish}(W\cdot\text{Flattened Output}+b)$$
32: Layer 10: Output Dense Layer
33: Apply a dense layer to match the original input shape
34: Reshape the output to $\mathbb{R}^{G\times G\times 3}$
35: Return: Predicted coordinates $\hat{D}$
```

Our AccelMD framework should have huge potential to reduce computing costs compared to executing the MD simulation using the brute-force method. For each MD simulation input (PDB files), our AccelMD framework will dynamically retrain the model according to the input files and ensure good prediction accuracy.

D. Self-adaptive Ability for Random PDB File Inputs

The preprocessing step of the AccelMD framework is designed to adapt any grid size (symbol 'g'), which is determined by the number of residues of the input protein structure. 'g' should be the minimal integer to guarantee g^2 is greater than or equal to the total number of residues from inputs.

E. An AccelMD Computational Example

This section provides an example to help understand the details of the AccelMD implementation. Assuming the input protein file has 239 residues and the starting timestep of MD simulation is t_0. The grid size $'g'$ should be $ceil(\sqrt{239}) = 16$. The *Preprocessing* stage reorganizes all atom's coordinates (x, y, z) into a 3D $(16, 16, 3)$ grid. The first Conv2D layer applies 32 filters of size 3×3, resulting in a $(16, 16, 32)$ output. This output calculation is as follows:

$$\text{Output}[i,j,k] = \text{Mish}\left(\sum_{m=0}^{2}\sum_{n=0}^{2}\sum_{c=0}^{2}\text{Input}[i+m, j+n, c]\cdot\right.$$
$$\left.\text{Kernel}[m,n,c,k] + \text{Bias}[k]\right) \tag{5}$$

The first *MaxPooling2D* layer downsamples the 3D $(16, 16, 32)$ output to $(8, 8, 32)$ by taking the max in each 2×2 window. The *BatchNormalization* layer normalizes the $(8, 8, 32)$ output to stabilize learning. The *Self-Attention* layer calculates the attention weights and output for each position:

$$\begin{aligned}\text{Scores}[i,j,h] &= \frac{Q[i,j,h]\cdot K[i,j,h]^T}{\sqrt{d_k}}\\ \text{Weights}[i,j,h] &= \text{softmax}(\text{Scores}[i,j,h])\\ \text{Output}[i,j,h] &= \sum_{p,q}\text{Weights}[i,j,p,q,h]\cdot V[p,q,h]\end{aligned} \tag{6}$$

The second *MaxPooling2D* layer downsamples the $(8, 8, 64)$ output to $(4, 4, 64)$ by taking the max in each 2×2 window. The second *BatchNormalization* layer normalizes the $(4, 4, 64)$ output. The Flatten layer reshapes the $(4, 4, 64)$ output to an 1D vector with 1024 elements. The fully connected *Dense* layer applies 128 units with Mish activation:

$$\begin{aligned}\text{Dense Output} = \text{Mish}(&\text{Weights}\cdot\text{Flattened Output}\\ &+\text{Bias})\end{aligned} \tag{7}$$

The second fully connected *Dense* layer transforms the 1D vector with 128 elements to an 1D vector with 768 elements, matching the total element number of preprocessing output. Finally, the *Reshape* layer unfolds the 1D vector with 768 elements back to the 3D $(16, 16, 3)$ grid, representing the predicted atomic coordinates of the protein structure.

IV. Evaluation Methodology and Experimental Results

To validate the AccelMD framework, we define '*prediction accuracy*' as the structural similarity between the coordinates predicted by AccelMD and the '*ground truth*' coordinates generated by a full, conventional MD simulation run for the same duration. While direct comparison with wet-lab

TABLE I: Mean Absolute Error (MAE) Comparisons for Multiple CNN Models with Different PDB File Inputs

PDB File Inputs	AccelMD	Resnet50	Alexnet	Lenet	VGGNet	GoogleNet	DenseNet
8sk7 (num_res=1653)	**0.0041**	0.0237	0.0128	0.0131	0.0236	0.0052	0.0351
8xj3 (num_res=476)	**0.0052**	0.0067	0.0086	0.0096	0.0363	0.0055	0.0289
2ddg (num_res=219)	**0.0055**	0.0134	0.0080	0.0093	0.0381	0.0064	0.1900
9bzb (num_res=270)	**0.0062**	0.0393	0.0077	0.0086	0.0342	0.0065	0.0087
8sk8 (num_res=2976)	**0.0046**	0.0096	0.0162	0.0161	0.0159	0.0059	0.0588
7vr1 (num_res=1256)	**0.0058**	0.0274	0.0155	0.0121	0.0272	0.0059	0.4381
5ghd (num_res=97)	**0.0084**	0.0162	0.0110	0.0115	0.0209	0.0104	0.2064
6os9 (num_res=1285)	**0.0066**	1.7600	0.0137	0.0161	0.0263	0.0068	0.0278

TABLE II: TM-score Comparisons for Multiple CNN Models with Different PDB File Inputs

PDB File Inputs	AccelMD	Resnet50	Alexnet	Lenet	VGGNet	GoogleNet	DenseNet
8sk7 (num_res=1653)	0.99990	0.99997	0.99990	0.99999	0.99997	0.99990	0.99990
8xj3 (num_res=476)	0.99976	0.99960	0.99933	0.99916	0.98746	0.99972	0.98924
2ddg (num_res=219)	0.99948	0.99128	0.99883	0.99839	0.97345	0.99926	0.96245
9bzb (num_res=270)	0.99938	0.96898	0.99906	0.99885	0.98139	0.99939	0.99928
8sk8 (num_res=2976)	0.99932	0.99932	0.78037	0.99958	0.99932	0.99992	0.98994
7vr1 (num_res=1256)	1.00000	0.99996	0.99999	0.99999	0.99996	1.00000	0.84667
5ghd (num_res=97)	0.99997	0.99994	0.99902	0.99945	0.99914	0.99940	0.92730
6os9 (num_res=1285)	0.99999	0.88930	0.99998	0.99998	0.99999	0.99999	0.99993

experiments was beyond the scope of this study, the accuarcy and atomic structure similarity were compared with the state of the art MD simulation tools. For validating the results of the proposed framework we calculated Mean Absolute Error (MAE) and TM-score, which measure the deviation between the AccelMD-predicted structures and the ground-truth simulated structures at equivalent points in time. The TM-score ranges from 0 to 1, where 1 indicates a perfect match and a lower score denotes less similarity. The TM-score calculation is as follows:

$$\text{TM-score} = \max\left(\frac{1}{L_{\text{target}}}\sum_{i=1}^{L_{\text{common}}}\frac{1}{1+\left(\frac{d_i}{d_0}\right)^2}\right) \tag{8}$$

L_{target} is the length of the target protein structure; L_{common} is the number of aligned residues; d_i denotes distance between the i-th pair of aligned residues; d_0 is a scale factor, calculated as $d_0 = 1.24 \cdot \sqrt[3]{L_{\text{target}} - 15} - 1.8$; The formula of MAE is available in [17].

Our experimental results proves that the AccelMD framework significantly outperforms other popular neural network models in both MAE and TM-score across various PDB file inputs with different residues. Other neural network models we evaluated contains *Resnet50* [18], *Alexnet* [19], *Lenet* [20], *VGGNet* [21], *GoogleNet (Inception V1)* [22], and *DenseNet* [23]. Besides the prediction accuracy, we will also discuss the adaptability of AI models and evaluate how much speedup AccelMD can obtain when compared with the conventional MD approach. The input PDB files we employ for experiments including *8sk7*, *8xj3*, *2ddg*, *9bzb*, *8sk8*, *7vr1*, *5ghd*, and *6os9*, with residue counts ranging from *97* to *2976*.

A. Mean Absolute Error (MAE) Comparisons

Figure 5 and Table I compare the Mean Absolute Error (MAE, in Ångströms) of AccelMD against six other popular neural network models across proteins of varying sizes.Here 'num_res' means number of residues. MAE should be 0 for the perfect match, and larger MAE values indicate worse similarity and inaccurate prediction. Our AccelMD consistently achieved the lowest error with an average MAE of just *0.0058* Å, demonstrating its high accuracy in predicting dynamic protein structures. The next best model, GoogleNet, achieved a slightly higher average MAE of *0.0065* Å.

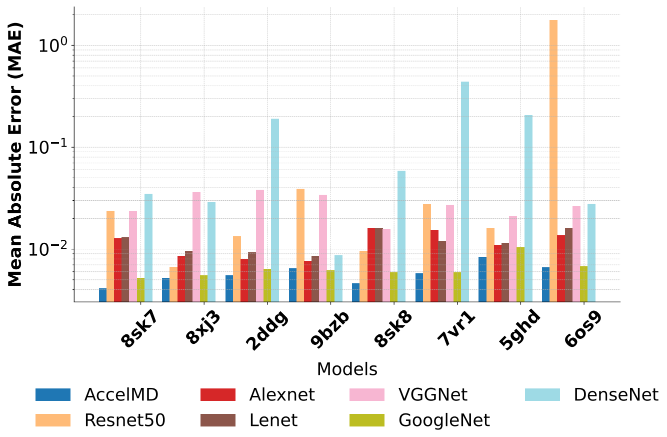

Fig. 5: Mean Absolute Error (MAE) Comparisons

Figure 6 shows the comparison of structural changes of the protein of PDB ID '7vr1' at various stages of molecular dynamics (MD) simulation with our AccelMD framework. The original structure represents the initial structure prior to any simulation. The fixed protein is the protein structure generated by 'PDBFixer' tool to fix missing residues and bonds. The simulated structure after 20 nanoseconds and 30 nanoseconds demonstrates the structural changes by conventional MD simulation. The predicted 20 nanosecond and 30 nanosecond structures, generated by our AccelMD model

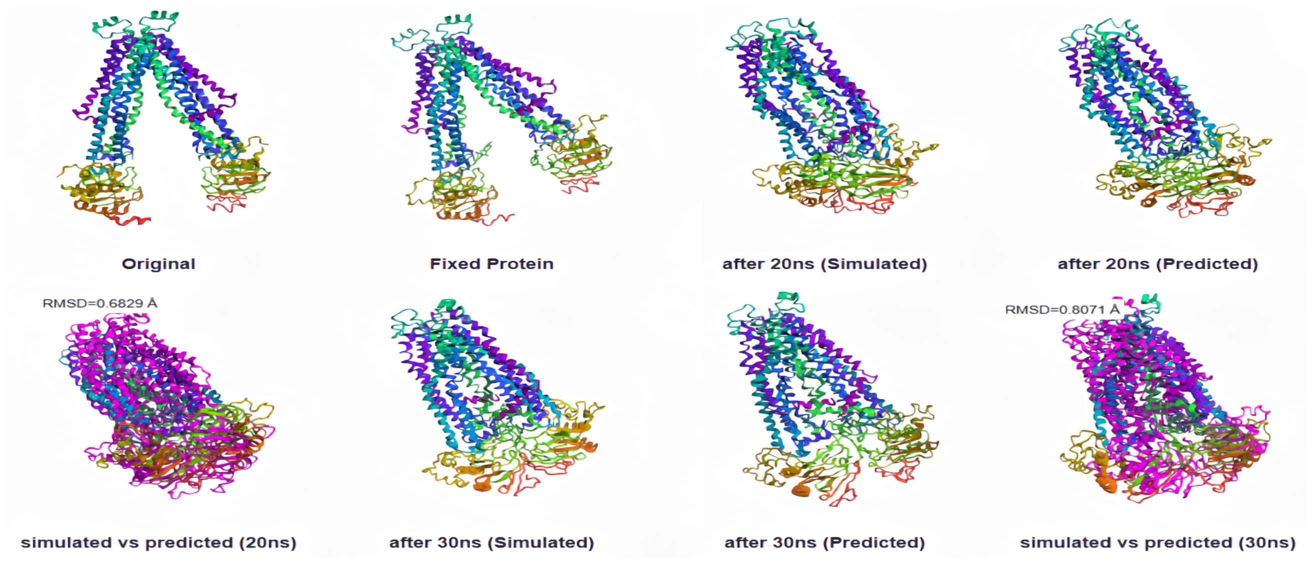

Fig. 6: Comparison of protein structural changes over molecular dynamics simulation and AccelMD prediction for PDB file '7vr1'. **Top row** (left to right): the original protein structure, fixed protein (fixing missing residues and bonds), the simulated structure after 20 nanoseconds, and the predicted structure at 20 nanoseconds generated by AccelMD. **Bottom row** (left to right): the superposition of the simulated and predicted structures at 20 nanoseconds (RMSD = 0.6829 Å), the simulated structure after 30 nanoseconds, the predicted structure at 30 nanoseconds, and the superposition of the simulated and predicted structures at 30 nanoseconds (RMSD = 0.8071 Å).

trained only on the initial 20-nanosecond simulation data. The final image overlaps the simulated ground truth and the predicted structures at 30 nanoseconds with a RMSD of 0.8071 Å. In protein structures comparison an RMSD value below 2 Å generally indicates high similarity. This relatively low RMSD value underscores the accuracy and reliability of the AccelMD framework.

B. TM-score Comparisons

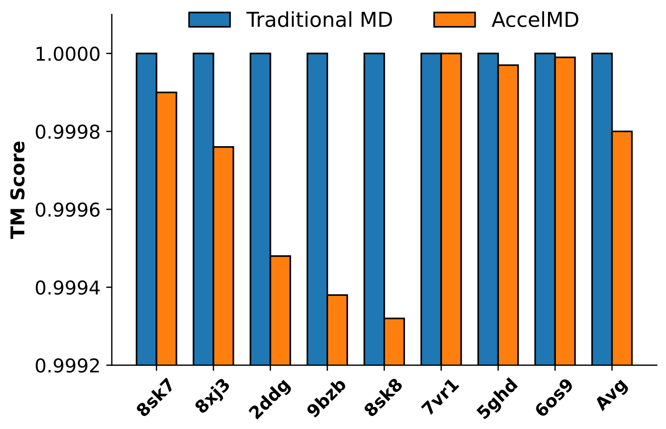

Fig. 7: TM-score Comparisons

Similarly, Figure 7 and Table II show the TM-scores of the same set of AI models with multiple PDB input files. TM-scores of conventional MD simulations (OpenMM) are normalized as '1.0'. Most of the models displayed instability in TM-scores except for *GoogleNet* and *AccelMD*, which achieve almost the same accuracy (from a TM-score perspective) when compared to traditional MD, and their average TM-score across all tested PDB files is *0.99976* and *0.99980*, respectively. Combining the observations from Section IV-A, only *GoogleNet* and *AccelMD* can accurately predict the protein structure in the MD simulation.

C. Adaptability to Random/Unknown Protein Inputs

Adaptability to all random or unknown inputs (PDB files) is also an essential metric to assess the model's feasibility in accelerating MD simulations. Without our self-adaptive pre-processing layer most of the other beanchmarking CNN based models fail to process the random protein inputs.*GoogleNet* and *DenseNet* appeared to have negative dimension sizes during pooling operations, particularly when handling smaller sized protein. *VGGNet* struggled with its deeper layers in convergence for PDB files with fewer residues. In contrast, AccelMD has excellent adaptability across all tested PDB files and can automatically adapt to complicated protein structures. This flexibility ensures that *AccelMD* seamlessly adapts to random PDB files. Therefore, although *GoogleNet* can also accurately predict protein structures, considering the adaptability and flexibility to unknown inputs, *AccelMD* is the only model that can effectively speed up general MD simulations.

D. Total Processing Time Comparisons

The main purpose of developing the AccelMD framework is to reduce total MD simulation processing time and lower computing costs while preserving great prediction accuracy. The prediction accuracy of the AccelMD framework has been verified in Section IV-A and Section IV-B. Figure 8 compares the total execution time to finish MD simulations between traditional MD (OpenMM) and AccelMD with multiple protein inputs. The runtime comparison clearly demonstrate that *AccelMD* significantly reduces the computation time compared to traditional MD simulation tools across all tested PDB files. The *47.59*X speedup on average (max: *63.78*X, min:

*30.61*X) proves AccelMD's efficiency in processing large protein structures.

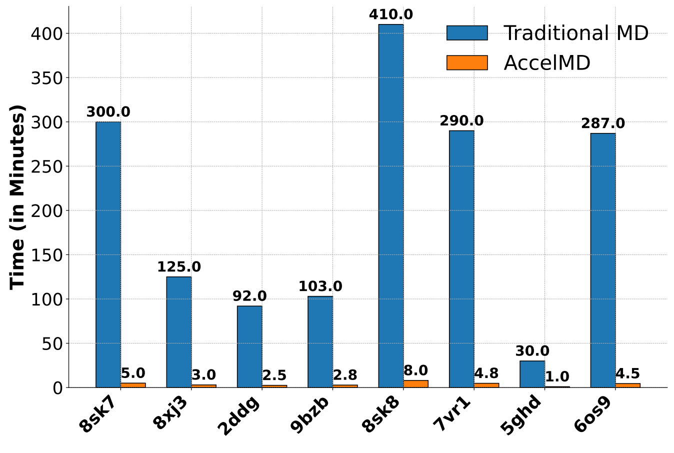

Fig. 8: MD Simulation Processing Time Comparisons

Besides the simulation time, the AccelMD approach can significantly lower the computational resource requirement to execute MD simulations. Conventional MD simulations typically execute on High-Performance Computers (HPC) with high-end GPUs to lower intolerable simulation time. In contrast, AccelMD can achieve similar tasks even using standard personal computers, showcasing its efficiency in resource utilization and the potential of being widely usable.

Overall, compared with regular MD simulations, the AccelMD framework can significantly reduce MD simulation processing time for all tested protein inputs while maintaining excellent prediction accuracy.

V. Conclusion

This paper introduces the lightweight AccelMD framework, designed to overcome the heavy computational cost of traditional Molecular Dynamics (MD) simulations. Utilizing the outputs from a short, initial simulation, our framework learns to predict the long-term dynamic structures of proteins by dynamically training a Convolutional Neural Network (CNN) with a self-attention mechanism. This method achieves a significant average speedup of *47.59*X while maintaining high predictive accuracy, with an average Mean Absolute Error (MAE) of *0.0058* and a TM-score of *0.99980*. A key feature is the self-adaptive preprocessing stage that allows AccelMD to handle diverse and unknown protein inputs automatically. Our systematic experiments demonstrate that this approach effectively reduces MD processing time, lowering the computational barrier for studying complex, long-timescale phenomena while maintaining runtime-accuracy tradeoff. The future work will focus on extending the framework to all-atom predictions along with self adaptive energy and forces for complex and larger molecular systems.

References

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

[2] N. Plattner and F. Noé, "Protein conformational plasticity and complex ligand-binding kinetics explored by atomistic simulations and markov models," *Nature communications*, vol. 6, no. 1, p. 7653, 2015.

[3] P. Eastman, M. S. Friedrichs, J. D. Chodera, R. J. Radmer, C. M. Bruns, J. P. Ku, K. A. Beauchamp, T. J. Lane, L.-P. Wang, D. Shukla *et al.*, "Openmm 4: a reusable, extensible, hardware independent library for high performance molecular simulation," *Journal of chemical theory and computation*, vol. 9, no. 1, pp. 461–469, 2013.

[4] S. Plimpton, "Fast Parallel Algorithms for Short-Range Molecular Dynamics," *Journal of Computational Physics*, vol. 117, no. 1, pp. 1–19, Mar. 1995.

[5] J. Abramson, J. Adler, J. Dunger, R. Evans, T. Green, A. Pritzel, O. Ronneberger, L. Willmore, A. J. Ballard, J. Bambrick *et al.*, "Accurate structure prediction of biomolecular interactions with alphafold 3," *Nature*, pp. 1–3, 2024.

[6] D. Hohl and R. Jones, "First-principles molecular-dynamics simulation of liquid and amorphous selenium," *Physical Review B*, vol. 43, no. 5, p. 3856, 1991.

[7] A. Ilari and C. Savino, "Protein structure determination by x-ray crystallography," *Bioinformatics: Data, Sequence Analysis and Evolution*, pp. 63–87, 2008.

[8] K. Inomata, A. Ohno, H. Tochio, S. Isogai, T. Tenno, I. Nakase, T. Takeuchi, S. Futaki, Y. Ito, H. Hiroaki *et al.*, "High-resolution multi-dimensional nmr spectroscopy of proteins in human cells," *Nature*, vol. 458, no. 7234, pp. 106–109, 2009.

[9] X. Benjin and L. Ling, "Developments, applications, and prospects of cryo-electron microscopy," *Protein Science*, vol. 29, no. 4, pp. 872–882, 2020.

[10] P. Thölke and G. De Fabritiis, "Torchmd-net: equivariant transformers for neural network based molecular potentials," *arXiv preprint arXiv:2202.02541*, 2022.

[11] K. Schütt, P.-J. Kindermans, H. E. Sauceda Felix, S. Chmiela, A. Tkatchenko, and K.-R. Müller, "Schnet: A continuous-filter convolutional neural network for modeling quantum interactions," *Advances in neural information processing systems*, vol. 30, 2017.

[12] O. T. Unke and M. Meuwly, "Physnet: A neural network for predicting energies, forces, dipole moments, and partial charges," *Journal of chemical theory and computation*, vol. 15, no. 6, pp. 3678–3693, 2019.

[13] P. Eastman, J. Swails, J. D. Chodera, R. T. McGibbon, Y. Zhao, K. A. Beauchamp, L.-P. Wang, A. C. Simmonett, M. P. Harrigan, C. D. Stern *et al.*, "Openmm 7: Rapid development of high performance algorithms for molecular dynamics," *PLoS computational biology*, vol. 13, no. 7, p. e1005659, 2017.

[14] P. D. Bank, "Protein data bank," *Nature New Biol*, vol. 233, no. 223, pp. 10–1038, 1971.

[15] O. D. Team, "Pdbfixer," https://github.com/openmm/pdbfixer, 2024, accessed: 2024-08-13.

[16] A. Vaswani, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[17] M. A. Error, "Mean absolute error," *Retrieved September*, vol. 19, p. 2016, 2016.

[18] B. Koonce and B. Koonce, "Resnet 50," *Convolutional neural networks with swift for tensorflow: image recognition and dataset categorization*, pp. 63–72, 2021.

[19] H. Ismail Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P.-A. Muller, and F. Petitjean, "Inceptiontime: Finding alexnet for time series classification," *Data Mining and Knowledge Discovery*, vol. 34, no. 6, pp. 1936–1962, 2020.

[20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[21] K. Simonyan, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[22] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[23] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.