Brian K. Ryu
CS 230 Project Milestone - bryu@stanford.edu

Future Prediction in Brownian Dynamics Simulations Using Deep Neural Networks

Team Member: Brian Kang Ryu (bryu@stanford.edu)

1 Introduction

Brownian dynamics (BD) is a branch of molecular dynamics (MD) that simulates the stochastic motion of Brownian particles to predict material properties. These dynamic simulations compute the time-evolution of the simulation system by iteratively computing forces acting on particles at each time step, integrating over time to find the displacement (change in position), and updating positions for each particle. The primary difference between BD and MD is the size of particles that are modeled by the simulation; while MD precisely computes the dynamics of every molecule and achieve system sizes up to tens of nanometers, BD coarse-grains the molecules and simulates the stochastic Brownian motion of larger particles that each consists of many molecules to model systems of several micrometers.[1-2] Researchers often use BD simulations to obtain a fundamental understanding of soft and condensed matter such as suspensions, gels, and glasses.

Similar to MD, a principle challenge of BD simulations is the prohibitive computational work required to examine phenomena that occur at long time scales.[3-4] For example, simulating arrested phase separation and aging of gels may require $n_t = 10^{10}$ timesteps. Furthermore, these simulations often require system sizes of $N = 10^6$ particles to faithfully replicate macroscopic behavior, and the computational complexity of a single time step often scales as $O(N^2)$. The total resultant cost is the product of operations per time step and the number of time steps. Despite advances in computing capabilities, computational cost remains the biggest concern for researchers who utilize BD simulations. To ameliorate this challenge of computational cost, studies in the past decade have given attention to machine learning (ML) and deep learning (DL) techniques.[5-7] These studies aimed to tackle quantum mechanics or force field calculations to reduce the work per time step. However, little focus has been given to reducing the number of time steps of simulations, which still remains a challenge.

In this project, I utilize DL techniques to predict the future state (timestep) of a BD simulation system based on its current state. The goal is to bypass computing every time step for systems that undergo only small changes between each timestep. To the best of my knowledge, predicting a future timestep in BD/MD simulations via DL instead of explicit computation has not been done before and there is a paucity of previous literature. Using the LAMMPS molecular dynamics software,[8] I have generated a dataset comprising 14,400 simulation *snapshots* (files with a list of all particles positions at a given time step for a simulation), which undergo slow non-equilibrium processes that are mildly stochastic (Brownian) yet largely deterministic. A combination of ResNet and CNN architectures will be used to predict the positions of particles at a future time. Because this work explores a novel task and realm of applications of DL in BD research, this project will mainly focus on implementation of various neural network architectures and compare performance to rationalize which architecture performs best.

2 Details on dataset

The data set contains 14,400 LAMMPS *trajectory files* (.lammpstrj) or simulation *snapshots* that contain information pertaining to the simulation system.[8] Typical trajectory files used for scientific research contain information regarding the current timestep, size and number of particles in the simulation box, as well as information of each particle such as a unique particle ID, type (e.g. H_2O or CO_2), (x,y,z) coordinates, linear and angular velocities, charge and etc.

For this work, I use trajectory files from simulations that undergo deterministic phase separation with small stochastic Brownian particle displacements. The data files will contain minimal information, containing particle IDs, types, and (x,y,z) positions as floating point numbers.

These snapshots can be rendered as images for qualitative visual inspections as shown, for example, in *Figure 1*. Each .lammpstrj file contains two snapshots in a single file that are used as inputs and labels, and are 2×10^5 timesteps separated apart. Furthermore, the number of particles is set to be constant (8,788) to limit unnecessary computational work in developing and testing various architectures. It is noted that large-scale scientific research projects may involve simulation systems containing millions of particles.

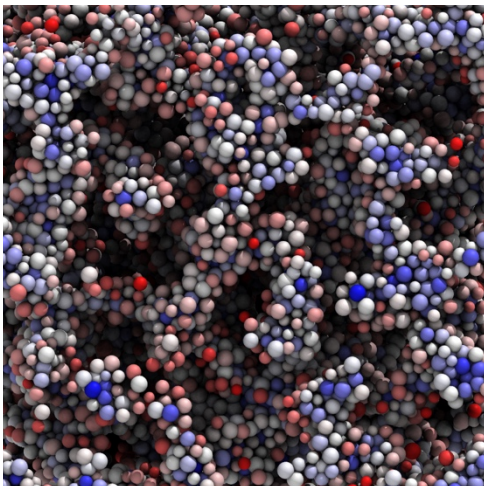

Figure 1: Simulation image rendered from a snapshot of a phase-separating colloidal system. Particle color indicates the number of neighbor contacts. Red: no contacts; blue: many contacts

3 Approach

3.1 Preprocessing

I have preprocessed the data set prior to usage for training. The preprocessing is done by normalizing all (x,y,z) coordinates by the simulation box size and then shifted by 0.5 so that most coordinate values lie within the range [-1, 1].

3.2 Network Architectures

As aforementioned, the prediction of entire simulation states in MD/BD simulations is a novel task with no literature precedent. Therefore, there exists no empirical intuition on which neural network architecture is most suitable for the given task. Hence, I rationalize the approach based on the physical processes modeled in the simulation.

First, the future state of the simulation predominantly depends and on the current state. Most physical processes do not go extensive structural changes within small windows of times, and particles may undergo only small displacements within $O(10^5)$ timesteps. Therefore, it is reasonable for the neural network to utilize a residual connection that allows full utilization of initial positions from the input layer via a skip connection, while the main connection contains hidden layers that compute the displacement, as illustrated in *Figure 2*.

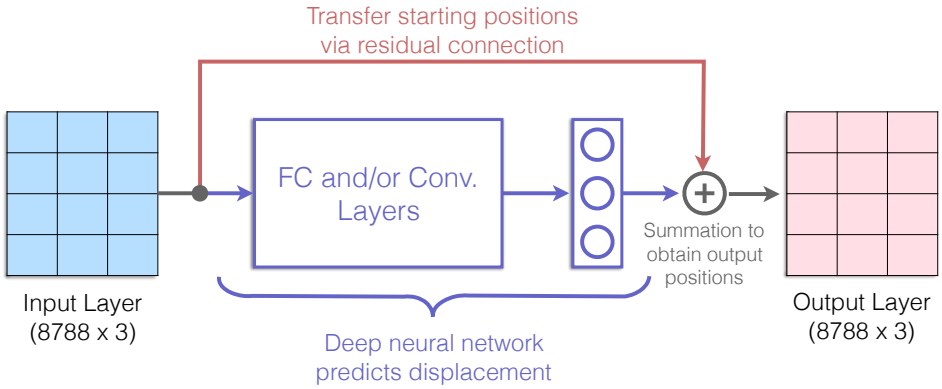

Figure 2: Schematic of residual network where the main path computes particles' displacement, which is added to the initial positions transferred via the skip connection

Furthermore, the displacement of particles in simulations with attractive particles heavily depend on number of interacting, or nearby particles. Particles with fewer neighbors (e.g. red particles in *Figure 1*) move freely under Brownian motion, whereas particles with many neighbors (e.g. blue particles in *Figure 1*) are bound to its neighbors. This idea of significance of neighboring

units is similar to that of CNN, where convolutional filters are employed to isolate the effect of surrounding pixels. Hence, I will incorporate convolutional layers of various filter sizes to consider the impact of neighboring particles in the main pathway of the residual network (*Figure 2*) that will compute displacements.

For the loss function and evaluation metric, both mean-squared error (MSE) and mean-absolute error (MAE) are suitable options because the output layer contains 8788×3 floating point values. These quantities are adequate metrics since it is possible to compute these quantities in simulations over the 10^5 timesteps to quantify Bayesian error. As of now, MSE was chosen as the evaluation metric.

3.3 Early Stage Results

I have constructed and trained three neural networks for 40 epochs, using the training and cross-validation (dev) sets. The Adam optimizer was used with no regularization and learning rate decay. The three architectures are: (i) FC_1: one fully connected hidden layer with 5,000 hidden units, (ii) FC_5: five fully connected hidden layers with 3,000 to 5,000 hidden units each, and (iii) $ResNet_1$: a network resembling *Figure 2* with four hidden layers with 4,000 to 5,000 hidden units in the main path and a residual connection. In-depth studies of convolutional filters are yet to be completed. The MSE vs. Epoch plot using two different learning rates is shown in *Figure 3*.

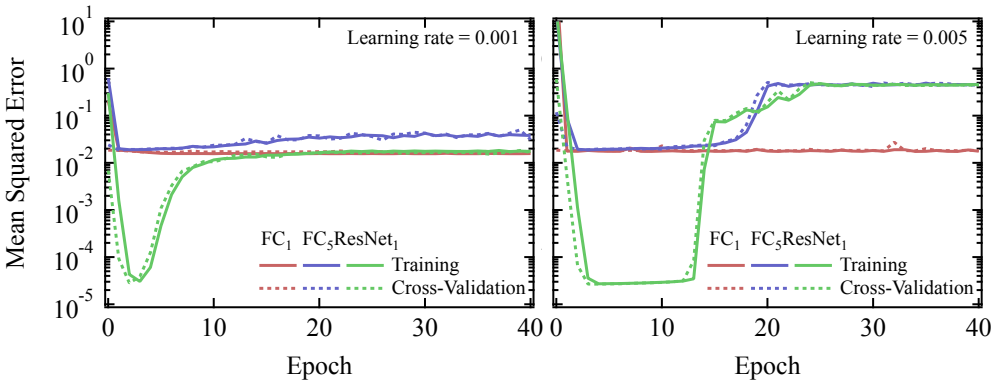

Figure 3: Mean-squared error vs. epoch plots from three neural networks with two different learning rates ($\alpha = 0.001$ and 0.005). The architectures are: (i) FC_1: one fully connected hidden layer, (ii) FC_5: five fully connected hidden layers, and (iii) $ResNet_1$: one residual block with four fully connected hidden layers.

As rationalized, we see that the residual connection (green in *Figure 3*) significantly improves the neural network performance with MSE values several orders of magnitude less than plain networks. Over many epochs, however, the MSE of the residual network increases to levels of plain networks, suggesting that further optimization of the neural network is required.

3.4 Remaining Work

The implementation discussed in section 4.3 is clearly preliminary. Going forward, I will first further explore neural network architectures with convolutional layers inside residual blocks, while potentially incorporating several residual blocks. Multiple filter sizes will be surveyed to identiy the architecture with the best performance. Then, optimization hyperparameters will be decided, focusing on regularization. Because the particle displacements must be small, L2-regularization, which favors small weights and therefore small activations, should deliver promising outcomes. Finally, hyperparameters will be optimized for fine tuning. Along with neural network optimization, an in-depth quantification of Bayes error will be quantified using simulations with different random seeds.

4 References

[1] Chen, Jim C., and Albert S. Kim. "Brownian dynamics, molecular dynamics, and Monte Carlo modeling of colloidal systems." *Advances in colloid and interface science* 112.1-3 (2004): 159-173.
[2] Erban, Radek. "From molecular dynamics to Brownian dynamics." *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 470.2167 (2014): 20140036.
[3] Zia, Roseanna N., Benjamin J. Landrum, and William B. Russel. "A micro-mechanical study of coarsening and rheology of colloidal gels: Cage building, cage hopping, and Smoluchowski's ratchet." *Journal of Rheology* 58.5 (2014): 1121-1157.
[4] Anderson, Valerie J., and Henk NW Lekkerkerker. "Insights into phase transition kinetics from colloid science." *Nature* 416.6883 (2002): 811.
[5] Noé, Frank. "Machine Learning for Molecular Dynamics on Long Timescales." *arXiv preprint arXiv:1812.07669* (2018).
[6] Mardt, Andreas, et al. "VAMPnets for deep learning of molecular kinetics." *Nature communications* 9.1 (2018): 5.
[7] Sarma, Sankar Das, Dong-Ling Deng, and Lu-Ming Duan. "Machine learning meets quantum physics." *arXiv preprint* arXiv:1903.03516 (2019).
[8] Plimpton, Steve. "Fast parallel algorithms for short-range molecular dynamics." *Journal of computational physics* 117.1 (1995): 1-19.