CS230

# Future Prediction in Brownian Dynamics Simulations Using Deep Neural Networks

**Brian K. Ryu**
Department of Chemical Engineering
Stanford University
bryu@stanford.edu

## Abstract

Computational cost of large and lengthy Brownian Dynamics simulations is a principle challenged faced by scientific researchers. In this work, I utilize deep residual networks to alleviate the computational cost of dynamic simulations by circumventing the computation of every time step and predicting future states of a simulation system. I present qualitative and quantitative evaluations of predictions, which are able to replicate the accuracy of simulations with a system size of 8,788 particles. This novel method for reducing the cost of simulations sets a foundation for a new avenue of deep learning application in the research of physical sciences.

## 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 a simulation system by iteratively computing forces acting on particles at each time step, integrating over time to find the displacement (changes in positions), and updating positions of particles. The primary difference between BD and MD is the size of particles that are being modeled; while MD precisely computes the dynamics of every molecule to model small systems ($< 10$nm), BD coarse-grains molecules into larger particles and simulates the Brownian and deterministic forces on larger particles that comprise many molecules to model larger systems ($< 1$mm).[1-2] Researchers use BD simulations to obtain understandings 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 the lifetime of a gel may require $O(10^{10})$ time steps. Furthermore, these simulations often require system sizes with $N = 10^6$ particles to faithfully replicate macroscopic behavior of materials where 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 continuous advances in computing power, computational cost remains the biggest concern for researchers who utilize BD simulations.

In this project, I utilize deep learning (DL) techniques to predict the future state (timestep) of BD simulations 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 in literarture. Using the LAMMPS molecular dynamics software,[5] I have generated a dataset comprising 14,400 simulation snapshots, which undergo slow non-equilibrium processes that are mildly stochastic (Brownian) yet largely deterministic. The input to my algorithm is a “simulation trajectory" file – a list of (x, y, z) positions of all particles at a given time step. I use ResNet

CS230: Deep Learning, Spring 2019, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

architectures to output a predicted simulation trajectory at a future time. Because this work explores a novel realm of application of DL in BD research, I have focused on the implementation of several neural network architectures and compared performance to rationalize which architecture performs best.

## 2 Related work

Several works in literature have attempted to develop machine learning (ML) and deep learning techniques to aid MD/BD simulations. One prominent category of DL for MD is in biomedicine; deep learning is beginning to impact biological research by integrating large data and molecular informatics for drug discovery and design.[6-8] Supported by rapidly growing genomic and proteomic sequence data, rational drug design via deep learning has shown significant improvement in the past few years and has yet realized its full potential. Another important field of DL application for physics and MD simulations is replacing computation-heavy density function theory (DFT) calculations of quantum mechanical equations with deep neural networks to reduce the computational work per time step in MD simulations.[9-10] These DL techniques have shown considerable amount of performance improvement, achieving upto a factor 100 improvement in performance, without significant loss in accuracy. Very recently, DL techniques for areas such as chemical reaction kinetics and computational structural biology have emerged,[11] but little attention has been given to actually reducing the number of time steps in simulations, which still remains a challenge. Hence, this work will attempt to tackle the unprecedented problem of how to bypass computing every single time step.

## 3 Dataset and Features

The data set contains 14,400 LAMMPS trajectory files (.lammpstrj) or simulation snapshots that contain information pertaining to the simulation system.[5] I have split the dataset into 12,000/1,200/1,200 as train/dev/test datasets. The dataset was created specifically for this project. Typical trajectory files used for scientific research include information such as: current timestep, size and number of particles in the simulation box, particle IDs, types (e.g. H2O or CO2), (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 contain 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 in Figure 1. Each .lammpstrj file contains two snapshots in a single file

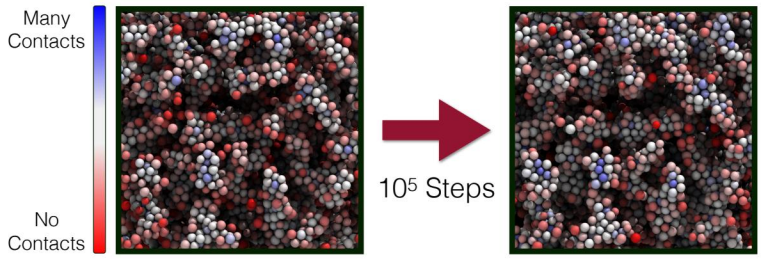


Figure 1: Images rendered from simulation snapshots undergoing slow phase separation, separated by $10^5$ time steps. Particle colors indicate number of contacts.

that are used as inputs and labels, and are $10^5$ timesteps separated apart. The number of particles is set to be constant (8,788) to limit the scope of project to developing and testing various architectures. Future work will extend the models in this work for larger systems and variable number of particles. The dataset preprocessed by normalizing positions with respect to the simulation box size so that all (x, y, z) particle positions range from 0 to 1.

## 4 Methods

Our model heavily utilizes residual connections inspired by the work by He *et al.*[12]

### 4.1 Model Description

The inputs and outputs of the model are 26364 floating point numbers corresponding to (x,y,z) positions of 8788 particles. The model contains several (up to seven) residual blocks, as shown

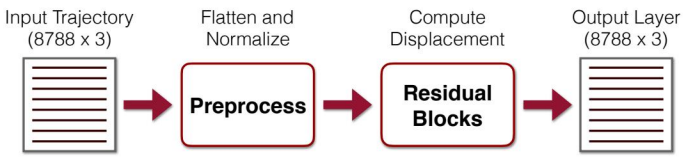


Figure 2: The structure of model architecture.

in Figure 2. Because both the input and output data are vectors of matching dimensions, I chose mean-squared error (MSE) as the loss function for the neural network. Since small changes in positions are expected, $L_2$ regularization was applied, which also helps prevent overfitting. The resultant cost function is:

$$J(\hat{y}, y) = \frac{1}{m} \sum_{i=1}^{26364} (\hat{y} - y)^2 + \lambda \sum_{j} w_i^2 \tag{1}$$

where $y$ is the label, $\hat{y}$ is the predicted output positions, and $\lambda$ is the regularization parameter.

### 4.2 Residual Block

The output activations of this neural network predicts an updated position based on the input list of positions. Hence, the future state predominantly depends on the current state and a residual block may take advantage of the nature of this computation. As depicted in the architecture of Figure 3,

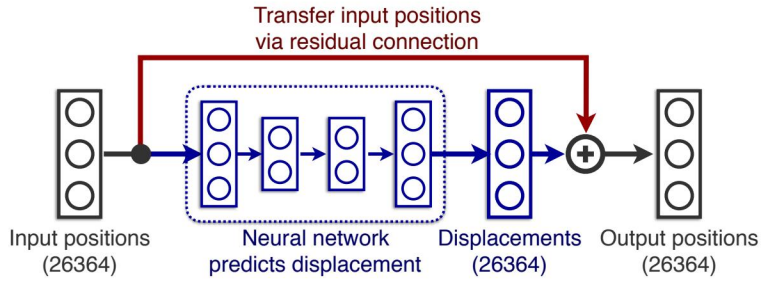


Figure 3: The schematic and rationale of a residual block.

the main path computes the particles' displacement whereas the initial position is transferred via a residual connection. The output position is predicted as the sum of initial positions and displacements. This architecture trains the neural network to learn to compute displacements between timesteps, which is similar to actual computations done in MD/BD software and thus achieves significantly better results compared to fully connected, plain networks.

## 5 Results and Discussion

### 5.1 Architecture Comparison

To evaluate the efficacy of neural networks in achieving the goals of this project, plain and residual networks of varying depths were examined. After exploring a range of hyperparameters, I used a learning rate $\alpha = 0.00005$, regularization parameter $\lambda = 0.00001$, and mini-batch size 512 to achieve maximum accuracy. The resultant training progress may be slow, but priority was given to reducing the loss; the ability to produce accurate results was preferred over small improvements in training time. Small perturbations in learning rate and regularization parameter near these values did not further improve accuracy and training rate beyond the current rate, suggesting a local minimum in training accuracy and rate. The model specifications and results are summarized in Table 1. Training progresses via validation loss as a function of epochs using these architectures are shown in Figure 4.

As seen in Table 1, Residual networks perform significantly better in predicting the future positions of particles than plain networks. As rationalized previously the superior performance is due to the nature of predictions made by the network; plain networks must learn to predict positions in addition

| Model | Type | Number of Layers/Res. Blocks | Average # of Nodes per Layer | Training Loss | Validation Loss | Test Loss |
|---|---|---|---|---|---|---|
| FC-1 | Plain | 1 Layer | 5000 | 2.41e-03 | 2.35e-03 | 2.34e-03 |
| FC-5 | Plain | 5 Layer | 4000 | 2.81e-03 | 2.71e-03 | 2.74e-03 |
| Res-1 | Residual | 1 Block | 3500 | 2.61e-05 | 2.55e-05 | 2.48e-05 |
| Res-3 | Residual | 3 Blocks | 3000 | 2.60E-05 | 2.55e-05 | 2.48e-05 |
| Res-5 | Residual | 5 Blocks | 1600 | 2.60e-05 | 2.56e-05 | 2.49e-05 |
| Res-7 | Residual | 7 Blocks | 1300 | 2.60e-05 | 2.55e-05 | 2.48e-05 |

Table 1: Summary of model architectures and training outcomes. Each residual block contained four fully connected layers. MSE was used for loss function. Training was done for 60 epochs.

to displacements, which is a significantly difficult task compared to that of residual networks, which simply need to predict the displacements and adding the initial positions. Because the neural network achieves similar levels of accuracies between training, dev, and test sets, it is unlikely to be overfitting the training set.

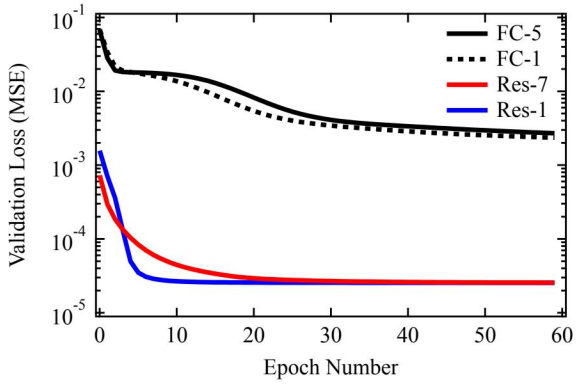


Figure 4: Validation accuracy for FC-1, FC-5, Res-1, and Res-7 networks over training epochs.

To qualitatively inspect and verify that accurate predictions are made, I have rendered an example pair of images from both predicted and simulated trajectories from the same input snapshot in Figure 5. From visual inspection, we see that the results are excellent; the two images in Figure 5(b-c) look qualitatively similar with no striking difference.

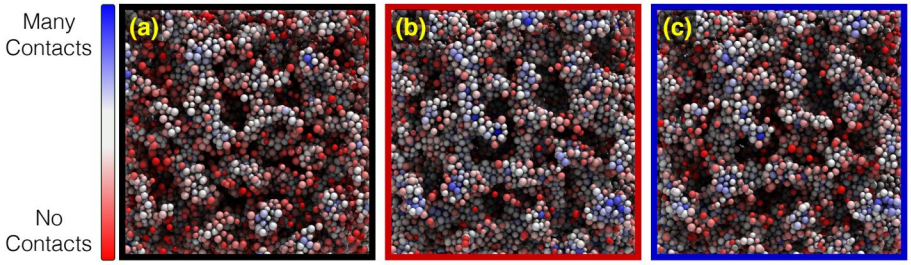


Figure 5: A comparison of images rendered from (a) input snapshot, (b) simulated (computed) snapshot after $10^5$ time steps, and (c) predicted snapshot. Particle color indicate number of contacts.

It is worth pointing out that approximately the same level of loss was achieved by all residual networks. A possible explanation for the same level of accuracy may be the size of the problem; predicting the positions of 8788 particles may simply not require a network with over 50,000 nodes. Figure 4 shows that even the shallowest Res-1 network achieves minimum loss in as early as 15 epochs. The advantage of deeper neural networks may become useful for systems with larger particle numbers up to $O(10^6)$, which is closer to the research-scale size used for large scale dynamic simulations used for studying macroscopic phenomena.

### 5.2 Randomness of Simulations as a Proxy for Bayes Error

The convergence of all errors from residual networks in Figure 4 naturally leads to the question of what is the level of theoretical minimum error. Simulations are always conducted by computers, and human-level error is unable to serve as a proxy for Bayes error in this task. As previously mentioned, however, the simulations in this work involve combinations of stochastic Brownian motion and deterministic interaction forces between particles, which result in largely deterministic yet mildly stochastic behaviors as a whole. Therefore, the extent of randomness can be quantified by repeatedly running simulations from a single snapshot with different random seeds, as described in Figure 6. These simulations will result in slightly different realizations of future states, whose MSE can be compared across different realizations to estimate the Bayes error as the mean degree of randomness.

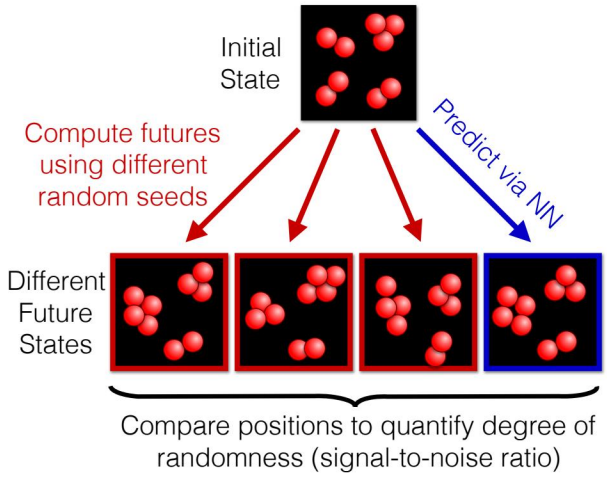


Figure 6: Schematic of Bayes error quantification methodology. Different random seeds in simulations result in slightly different futures.

To quantify the minimum error via this methodology, ten initial states were randomly chosen and two future states were respectively simulated and the MSE between each pair was computed. A summary of results are shown in Table 2. It is apparent that the neural network is able to achieve

| | Mean-Squared Error | |
|---|---|---|
| **Model** | **Mean** | **SD** |
| Simulations | 1.90e-05 | 8.90e-06 |
| Res-7 | 2.48e-05 | 1.11e-05 |

Table 2: Comparison of randomness in simulation with error from neural network-generated predictions. Error of neural networks is almost similar to randomness in simulations.

errors near the randomness of the simulations (i.e. Bayes error). This explains the identical level of accuracy obtained from networks with different levels of error and further drives the need to expand the applicability of the current technique for predicting futures of larger simulation systems.

## 6 Conclusions and Future Work

I have successfully developed a deep neural network for predicting future states of Brownian Dynamics simulations based on a current state. I have discovered that utilizing a residual network architecture to focus on displacement significantly improves accuracy with errors close to the Brownian noise of simulations. This performance is driven by allowing the neural network to focus on computing displacements to update the initial position via residual connections.

Future work should focus on visualization and interpretation of what and how each residual block is computing displacement. For generalization of this task, future work should also extend the current architecture for larger and variable number of particles for practical applicability in computational research environments. Finally, the possibility of recursive aging should be investigated for practical use as well; the ability to repeatedly feed output predictions into the input layer to make predictions of even further futures will significantly widen the usefulness of the current methods.

## 7 Contributions and Acknowledgements

This work was solely completed by Brian K. Ryu for completion of the final project of CS 230. I would like to acknowledge the authors of [3] for the proprietary code for generation of the dataset. I would also like to acknowledge Ahmad Momeni as well as the CS 230 teaching staff for helpful discussion and guidance for this project.

## 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] Plimpton, Steve. "Fast parallel algorithms for short-range molecular dynamics." Journal of computational physics 117.1 (1995): 1-19.

[6] Gawehn, Erik, Jan A. Hiss, and Gisbert Schneider. "Deep learning in drug discovery." Molecular informatics 35.1 (2016): 3-14.

[7] Lo, Yu-Chen, et al. "Machine learning in chemoinformatics and drug discovery." Drug discovery today 23.8 (2018): 1538-1546.

[8] Wainberg, Michael, et al. "Deep learning in biomedicine." Nature biotechnology 36.9 (2018): 829.

[9] Noé, Frank. "Machine Learning for Molecular Dynamics on Long Timescales." arXiv preprint arXiv:1812.07669 (2018).

[10] Wang, Han, et al. "DeePMD-kit: A deep learning package for many-body potential energy representation and molecular dynamics." Computer Physics Communications 228 (2018): 178-184.

[11] Goh, Garrett B., Nathan O. Hodas, and Abhinav Vishnu. "Deep learning for computational chemistry." Journal of computational chemistry 38.16 (2017): 1291-1307.

[12] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.