# SIMPLE: Meta-Learning Conservation Laws for End-to-End Molecular Dynamics Simulation

Jae-Won Chung[1], Dhiraj Maji[1], Vatsala Prasad[1], and Charlotte Zhao[1]

[1]University of Michigan
{jwnchung,dmaji,vatsala,zshiqi}@umich.edu

**Abstract**

Molecular Dynamics (MD) simulation, though a powerful tool with a vast variety of applications, has a huge computational overhead. In order to accelerate MD simulations, we propose an end-to-end deep learning approach, where we incorporate an unsupervised loss function to encode physical conservation laws as inductive biases into our model. We create our own dataset with 1,625 MD simulation trajectories and train/evaluate our proposed model on it. Experiment results show that our end-to-end DL framework accelerates simulation by 85x, and we discuss directions for improvement in detail.

## 1 Introduction

Molecular dynamics (MD) simulations have been widely used for studying classical many-body systems, whose behaviors obey the laws of classical mechanics. One particular advantage of molecular dynamics is that it closely mimics real experiments and provides microscopic details that are inaccessible in real experiments. An MD simulation proceeds as follows: a system of interest is chosen, and then measurements are made to quantify how the system evolves from initial conditions. Here, the evolution of the system is derived by solving Newton's equations, which has proven to be an excellent approximation for a variety of systems [4].

However, one limitation of MD simulation is that it is significantly slower compared to real experiments. To capture the rare yet crucial events that occur in real experiments, multiple replicas of the same initial condition (e.g., temperature and box dimensions) are needed. Thus it requires an enormous amount of computing power for MD simulations to sufficiently sample the configuration space of interest.

There have been constant efforts in enhancing the sampling efficiency by combining MD simulations with other techniques, but some of them still suffer from high computational cost and others affect the dynamics of the simulated system due to their biased nature [7].

Motivated by such limitations, we explore the possibility of an entirely different path: a end-to-end deep learning approach. Our method, to the best of our knowledge, has never been attempted in the domain of MD simulation. Specifically, we draw an analogy between MD simulation and video next-frame prediction, in that both tasks aim to predict the state of the next time step based on one or more prior time steps. However, there are still crucial differences that preclude the direct application of techniques from video next-frame prediction works.

First, the property of particles at each time step is constrained by the governing law of physics, which imposes a certain structure onto system data that are not easily leveraged by normal video

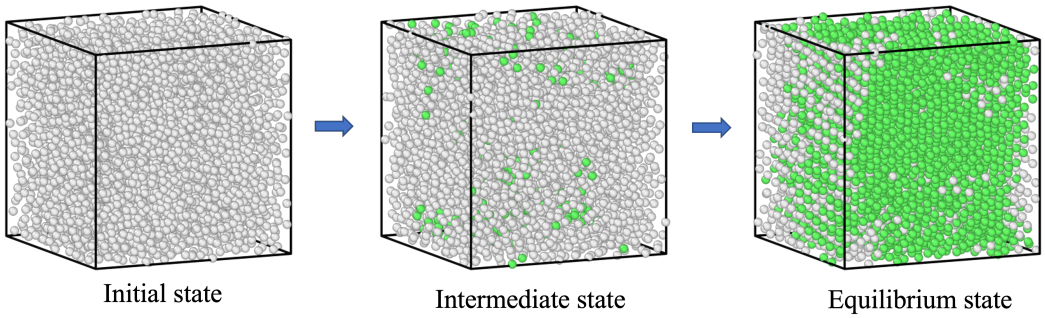


**Figure 1:** MD simulation of the self-assembly of a colloidal particle system. The initial state is disordered. Then particles begin to self-assemble into an ordered structure (colored by green). The transition process can be quantified using the fraction of particles in the ordered structure. At equilibrium, the fraction of ordered particles remains roughly constant.

next-frame prediction works. To address this challenge, we take inspiration from recent work in physics-inspired deep learning [5, 2]. That is, we seek to encode the implicit physical structure of the system data by adopting an additional unsupervised loss function that *tailors* [1] the next-frame predictor to ensure that its predictions obey the underlying physical conservation law. In essence, we are using physics as a strong inductive bias for our next-frame prediction model.

Second, the number of particles is exceedingly large in systems dealt in most MD simulations. Hence, direct applications of feed-forward neural operations or attention over particles result in an explosion of parameters. Moreover, compared with video frames (images) whose data format provides inductive bias in the form of spatial locality, allowing the use of efficient operations such as convolution, the state representation of MD simulation is extremely compact, consisting of an array of particle position and velocity vectors. Thus it is not trivial to use efficient DL operations on state matrices. Again inspired from physics, we reduce the size of the problem by only considering the `k` nearest neighbors of each target particle. While in theory every other particle affects the movement of a particle, it suffices as an accurate approximation to consider the effect of a few nearest particles.

Third, the size of the simulated system may vary. That is, systems with different numbers of particles may well be simulated by the user of the model, and the exact same physical conservation laws nonetheless apply regardless of the size of the system. Thus, naïvely designing the model so that they operate on all particles results in a model that is not *size-agnostic*, and the trained model can only be used for systems with a certain number of particles, significantly limiting the usefulness of the model. Our `k` nearest neighbor approach naturally solves this problem as long as the system size is larger than `k`, which is typical.

We believe our approach is beneficial compared to traditional MD simulation. First, by training our model with a limited number of states at selected time steps, we may accelerate simulation by arriving at the final state of simulation faster. Here, the opportunity for improvement comes from the fact that traditional MD simulation approaches are constrained to solving Newton's equations at infinitesimal time steps to approximate $dt$ numerically, while neural networks can learn to correctly map the state transition of longer time distances. Moreover, our model can be easily generalized to other systems that are suitable for MD simulations, since our framework allows users to implement their own conservation law across simulation time steps. Finally, we support *approximate conservation*. That is, we do not enforce exact conservation of quantities. In experiments, systems usually do not perfectly conserve physical quantities due to reasons such as heat dissipation. Our tailoring method allows for such fluctuations that more accurately reflect how experimental particle systems

behave.

We provide a summary of relevant deep learning theory in Section 2, and a detailed explanation of our proposed method in Section 3. Evaluation results are presented in Section 4. We discuss the limitations of our work and lay out future plans in Section 5, and conclude in Section 6.

# 2 Related Work

## 2.1 Tailoring and Meta-tailoring

Encoding inductive biases such as the symmetry of the data can greatly improve the generalizability of deep learning algorithms [9]. For example, the convolutional neural network (CNN), which has shown extraordinary performance in image classification, has translation equivariance built in: if the input is translated, the output is also translated. CNN is also translation invariant: an image of a cat is always classified as "cat" regardless of the position of the cat in the image. However, building in inductive biases can be very challenging, especially for exploiting symmetries, since they are either unknown without domain knowledge, or difficult to code in.

Popular approaches for encoding inductive biases include data augmentation and adding auxiliary losses to the main task loss. However, data augmentation has poor generalizability, and with auxiliary losses, the model is optimized on a loss different from the primary objective. Moreover, auxiliary losses are only used at training time, resulting in a generalization gap between training and testing [1].

To ameliorate these issues, [1] proposes the Tailoring framework, which trains a model with the regular supervised learning approach, and then uses an unsupervised loss function to fine-tune the model for each query data point during prediction time. Finally, the updated parameters are used to make the final prediction.

Building upon this idea, authors propose the Meta-tailoring [1] framework, which not only customizes the model at prediction time for query points, but also incorporates tailoring in the training process. In the two-step architecture of Meta-tailoring, the model is tailored to each training data point using the unsupervised tailoring loss in the inner loop, and then optimized with the main task loss in the outer loop. Then the model is updated again with the tailoring loss for each query point at prediction time, as in the tailoring framework. This approach allows for encoding inductive biases into a model without having to change the main task loss.

## 2.2 Noether Network and Meta-Learning Symmetries

One challenge of the meta-tailoring approach introduced above is designing the unsupervised loss function. In a lot of cases, inductive biases, or the hidden symmetries of data, cannot be easily described or encoded. In cases where it can be formulated, the user-specified loss function has very limited generalizability. To solve this issue, [2] proposes the Noether Network based on Noether's theorem, which can be rephrased as the following,

> For every continuous symmetry property of a dynamical system, there is a corresponding quantity whose value is conserved in time.

Thus the problem of encoding symmetries can be redefined as conserving the corresponding quantity, which is easily observed from data. The unsupervised loss function can then be designed to conserve meaningful quantities.

# 3 Proposed Method

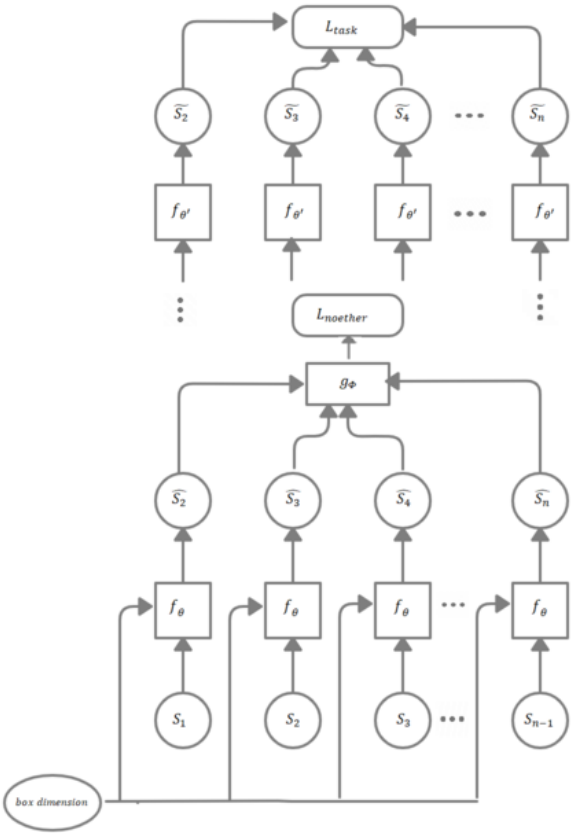


**Figure 2:** Training time forward propagation of SIMPLE. Teacher forcing is only used for training and not for inference.

In this section, we elaborate our proposed method SIMPLE: SIMulation via Physics-based LEarning. Find an overview of our training process, together with the models, in figure 2. Section 3.1 explains the precise definition of our task and the data involved. Then, section 3.2 goes through the models used, section 3.3 the training process, and section 3.4 the inference process.

## 3.1 Task and Data

The MD simulation task we study in this work is a process of figuring out the *state* of all particles in a certain particle system for each time step, where state is defined as the 3D position and velocity of each particle in the system. Hence, the state of a particle is completely described by a 6-dimensional vector. Since in MD simulation, the number of particles ($N$) is always fixed across time, the state matrix for a time step is a matrix with $N$ rows and 6 columns, where the first three columns represent the position of each particle and the rest velocity.

One extra component of the MD simulation task we consider here is the dimension of the simulation box in which the particles reside. The particles are initialized with certain initial positions and velocities, and after some time, the dimension of the box is reduced to simulate the decrease in volume or increase in pressure, which drives the particles' behaviors.

We denote the dimension of the box $B_t$ and the state matrix $S_t$ at time step $t$.

### 3.2 Model

We aim to build a model that, given the previous time step's state matrix and box dimension, can predict the next time step's state matrix. Our next-frame predictor model (henceforth denoted as $f_\theta$ where $f$ is parametrized by $\theta$) first identifies the $k$ nearest neighbor particles for each input particle, and then applies a simple three-layer fully-connected neural network activated by Sigmoid to predict the next state of the particle. Drawing intuition from skip connections [6], $f$ predicts the *changes* in the particle's position and velocity for the next time step. Moreover, since position is translation invariant, we offset the positions of all neighbor particles by the position of the target particle before passing it into the network.

Note that unlike usual time series prediction models, we *do not* adopt RNN architectures for $f$. Rather, our next state predictor is memoryless. This is because we have a guarantee from physics that it is possible to predict the next state $S_t$ solely based on the previous state $S_{t-1}$.

### 3.3 Training

For an instance of simulation data, which includes $n$ many time steps, the model $f_\theta$ is given $B_t$ and $S_t$ and predicts $\hat{S}_{t+1}$. Then, all $\hat{S}_t$'s are separately input into the *quantity predictor* $g_\phi$, which can either be learned or not learned. The role of the quantity predictor function is to compute a physically conserved quantity from the state matrix, e.g. energy. With the quantities computed by $g_\phi$, $L_{noether}$ [2] simply penalizes the L2 distance between the $n$ quantities, forcing the quantities to be *conserved* across time steps.

After the value of $L_{noether}$ is obtained, we perform one SGD step with respect to $\theta$, obtaining the tailored weight $\theta'$. Finally, $f_{\theta'}$ predicts the state of each time step $\tilde{S}_{t+1}$ given the same inputs $B_t$ and $S_t$, and we back propagate from the actual task loss function $L_{task}$. We use mean L2 distance for the task loss for MD simulation. Our overall process of training was inspired by [1].

Note that our framework allows users to plug in a quantity predictor function easily, as long as the function is differentiable with respect to $\theta$. Moreover, we allow multiple quantity predictors to be used at the same time, i.e. the output of each predictor are concatenated and their L2 distance is penalized. We experiment with a learned quantity predictor (a three-layer fully-connected network activated by Sigmoid) and a fixed quantity predictor (a *temperature predictor* which ensures that the temperature of the system remains constant, as is the case for the simulation data we use).

### 3.4 Inference

While during training we have access to the ground truth states of each time step $S_t$, during evaluation, we only have access to $S_1$ and $B_t$ ($t \in [1, T]$). Note that $B_t$ is input by the user who wishes to guide the simulation by reducing or expanding the dimension of the box. For the first time step the inputs to $f_\theta$ are $S_1$ and $B_1$, and for every subsequent time step $t$ the inputs are $\hat{S}_t$ and $B_t$.

## 4 Evaluation

**Simulation Dataset** As deep learning models have never been used directly for simulating MD systems before, there is no standardized dataset we can use to train, evaluate, and test our model. We created a dataset of 1,625 MD simulations ourselves, which we use as the ground truth.

To make sure our model can generalize to different simulation conditions (i.e., temperature and box dimension), we created a dataset that contains 13 different temperatures and 25 different box

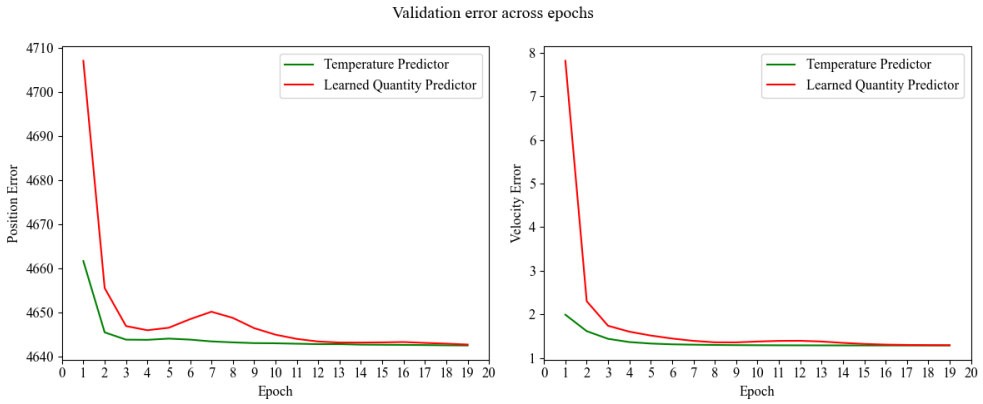


**Figure 3:** Validation errors of position (left) and velocity (right) for Temperature Predictor and Learned Quantity Predictor

dimensions. For each combination of temperature and box dimension, we generated 5 replicas, 3 of which were used for training, 1 for validation, and 1 for testing. For each replica, the positions and velocities of all particles are initialized using a random number generator. Each simulation example has around 250 frames (the length of the series is not fixed), which contains both a transition process driven by the underlying physics, and some frames of the final equilibrium state after the transition, where the positions and velocities only fluctuate by a small amount.

**Accelerating MD simulation** We compare the latency of traditional MD simulation and our DL model on the same Greatlakes GPU cluster. Note that HOOMD-blue [3], the traditional MD simulation framework we use, is also accelerated by GPUs. HOOMD-blue takes 230 seconds to reach the end of simulation. On the other hand, since our model learns to *jump* longer time distances, its inference latency for one simulation is 2.7 seconds, realizing a speed up of **85x**.

**Simulation accuracy** In Figure 3 we show the validation error[1] of two variants of our model: one with a temperature predictor and another with a learned quantity predictor, both with 10 neighbors in kNN. Error is measured by the L2 distance between the ground truth and the prediction, and we breakdown the errors in terms of position and velocity. The temperature predictor model converges faster than the learned quantity predictor model, but eventually both converge to the same error. This suggests that the learned quantity predictor (1) was able to identify temperature as the value that must be conserved across time and (2) system temperature is the most salient, if not only, non-trivial physical quantity that is conserved in this particle system.

**Error accumulation over time** In Figure 4 we show the position and velocity validation error over time after the first and the nineteenth epoch for the same batch of data, respectively. Error is noticeably lower after the model is trained, but in both cases, position error accumulates across time steps. This is because the model operates on the more and more inaccurate state matrices after each time step. However, the error being linear with respect to the step after training suggests that the per-step error of the state predictor is roughly constant. On the other hand, the velocity error of the nineteenth epoch decreases across time steps, suggesting that our model can accurately predict velocity.

[1]While we also have a test set, we deliberately choose to not evaluate on it because this model is still at an early stage. We discuss ways to improve the current model in Section 5 and leave testing as a next step.

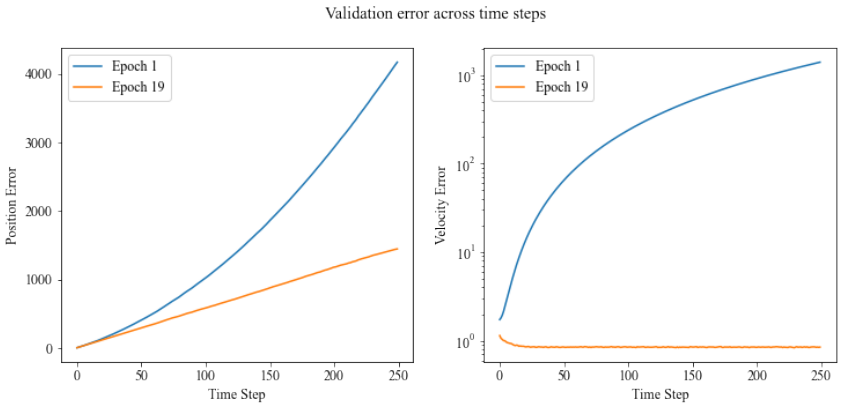


**Figure 4:** Position (left) and velocity (right) validation errors of the same batch of data across 250 time steps after one and nineteen epochs of training. Plotted for the learned quantity predictor model. The temperature predictor model shows similar trends.

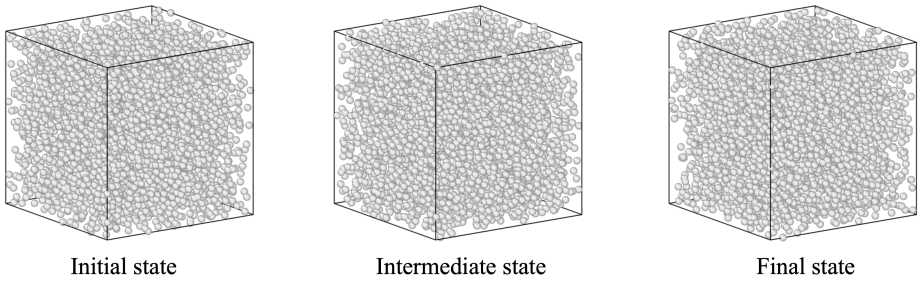


**Figure 5:** Predictions of an intermediate state and a final state made by our model with the Temperature Predictor given the initial state.

**Visualization of predicted states** To further inspect our results, we visualize the predictions made by our learned quantity predictor model for one validation sample. Selected intermediate and final frames are shown in Figure 5. We observe that the predictions for the intermediate state and the final state are very similar. This is the result of the model trying to *average* over the entire simulation trajectory. That is, the L2 loss function applied uniformly across all time steps are not enough to force the model to accurately predict the state matrix of each time step. Rather, it makes the model smooth out its predictions across time steps and still optimize the task loss. We discuss ways to improve this in Section 5.

## 5 Limitations and Future Plans

**Improving the dataset** *Sampling rate*, which is equivalent to the fraction of frames we sample from the entire simulation trajectory and feed to our model, is an important knob that trades off the amount of information between sampled frames and the speed up of the system with respect to traditional MD simulation. That is, too small a sampling rate will lead to no speed up, while too large a sampling rate will render the problem very difficult for the model. However, the current sampling rate was chosen arbitrarily. Thus a concrete next step is to coarsely sweep different sampling rates and observe the accuracy of our model.

In addition, many simulation trajectories in our current dataset contain more equilibrium states

than states in the transition process, which may bias our model to assume equilibrium for more time steps, which is also related to the *smoothing out* behavior of our model (§4). Therefore, more frames of the transition process are needed for the model to more accurately predict the transition.

Finally, our dataset contains simulations starting from vastly different initial conditions. At higher temperatures, the particles are more active and have higher velocities, which affect the ratio of frames in transition and at equilibrium. Therefore, scaling down the dataset and having the model learn from a limited number of initial conditions would improve its performance.

**Improving the model** The choice of $k$ in kNN is currently arbitrary. Yet, $k$ is an important parameter that determines how *far* the model will look when predicting the next state of each particle. The choice of $k$ must also consider (1) the system of interest, because the effective neighborhood of a particle must be larger for systems with particles that move faster, and (2) the sampling rate, requiring larger neighborhoods when the sampling rate is low.

In addition, we may augment the input data, such as adding the logarithm of velocities to the input to the quantity predictor, since it may be difficult for the learned quantity predictor to recover the conservation law at play if the conserved quantity is a non-linear combination of position and velocity.

Next, as shown by the visualization of our model's prediction, a naïve L2 error function across all time steps does not sufficiently force the model to predict accurate states for each time step. Loss functions based on descriptors of the state configurations, such as the Steinhardt order parameters [8] can be used as alternatives. More specifically, instead of naïvely penalizing the L2 distance between the position predictions with the ground truth, the loss function compares the order parameters computed using the positions.

Finally, due to the lack of time and hardware, we were not able to perform hyperparameter tuning for our model, which we leave as future work.

**Improving evaluation** The evaluation overall lacks ablation study. In order to fully motivate the need for meta-tailoring, it is necessary to compare our model with baselines without meta-tailoring, e.g. using the Noether loss as an auxiliary loss, and a simple network without Noether losses (that do not explicitly adhere to conservation laws).

Furthermore, comparing simulation latency with works that are not end-to-end (i.e., those that use DL as a component in the simulation process) would be interesting. Such systems are expected to be slower because they often require more frequent data copies between the host and GPU memory.

# 6 Conclusion

In this work, we have investigated the feasibility and gains of applying deep learning in an end-to-end fashion to Molecular Dynamics simulation. We collected a large dataset of 1,625 simulation trajectories and proposed a model based on meta-tailoring that explicitly tailors the next frame predictor such that it adheres to physical conservation laws. Evaluation results show that our model is able to significantly accelerate simulation and improve its predictions by learning from our dataset. Importantly, evaluation results reveal concrete steps towards improving our dataset and methodology.

## References


[1] F. Alet, M. Bauza, K. Kawaguchi, N. G. Kuru, T. Lozano-Perez, and L. P. Kaelbling. Tailoring: encoding inductive biases by optimizing unsupervised objectives at prediction time. 9 2020.

[2] F. Alet, D. Doblar, A. Zhou, J. Tenenbaum, K. Kawaguchi, and C. Finn. Noether networks: Meta-learning useful conserved quantities. 12 2021.

[3] J. A. Anderson, J. Glaser, and S. C. Glotzer. Hoomd-blue: A python package for high-performance molecular dynamics and hard particle monte carlo simulations. *Computational Materials Science*, 173:109363, 2020.

[4] D. Frenkel and B. Smit. Chapter 4 - molecular dynamics simulations. In D. Frenkel and B. Smit, editors, *Understanding Molecular Simulation (Second Edition)*, pages 63–107. Academic Press, San Diego, second edition edition, 2002.

[5] S. Greydanus, M. Dzamba, and J. Yosinski. Hamiltonian neural networks. *Advances in Neural Information Processing Systems*, 32, 6 2019.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[7] S. Hussain and A. Haji-Akbari. Studying rare events using forward-flux sampling: Recent breakthroughs and future outlook. *The Journal of Chemical Physics*, 152:060901, 2 2020.

[8] P. J. Steinhardt, D. R. Nelson, and M. Ronchetti. Bond-orientational order in liquids and glasses. *Phys. Rev. B*, 28:784–805, Jul 1983.

[9] A. Zhou, T. Knowles, and C. Finn. Meta-learning symmetries by reparameterization. 7 2020.

# A Author Contributions

- **Jae-Won**: Project idea, Proposal writing, Poster presentation, Report (Introduction, Proposed Method, Evaluation, Limitations and Future Plans, Conclusion), Implementation (Model architecture, training/inference logic, loss, experiment)
- **Dhiraj**: Proposal editing, Poster presentation and editing, Report (Abstract, Related Work, Evaluation), Implementation (`argparse`, experiment, tensorboard logging)
- **Vatsala**: Proposal editing, Poster presentation and editing, Report (Proposed method, Evaluation, Conclusion), Implementation (Data creation, data pipeline, experiment, tensorboard Logging)
- **Charlotte**: Proposal writing, Poster presentation and draft, Report (Introduction, Related work, Evaluation, Limitations and Future Plans, Conclusion), Implementation (Dataset collection, data pipeline, experiment, tensorboard logging)