Macromolecules

pubs.acs.org/Macromolecules Article

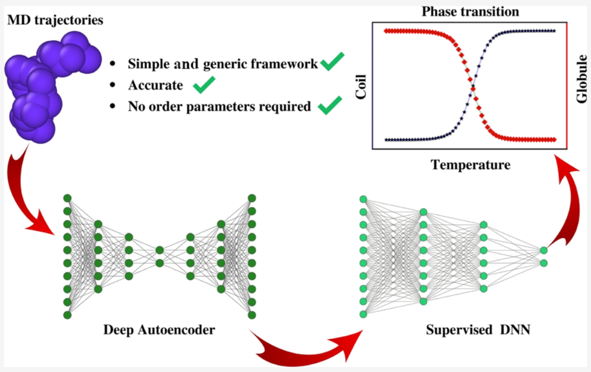
dPOLY: Deep Learning of Polymer Phases and Phase Transition

Debjyoti Bhattacharya and Tarak K. Patra*

Cite This: *Macromolecules* 2021, 54, 3065−3074 Read Online

ACCESS | Metrics & More | Article Recommendations | Supporting Information

ABSTRACT: Machine learning (ML) and artificial intelligence (AI) have remarkable abilities to classify, recognize, and characterize complex patterns and trends in large data sets. Here, we adopt a subclass of ML methods, viz., deep learning, and develop a general purpose AI tool—dPOLY—for analyzing molecular dynamics (MD) trajectories and predicting phases and phase transitions in polymers. An unsupervised deep neural network (DNN) is used within this framework to map a MD trajectory undergoing thermophysical treatment such as cooling, heating, drying, and compression to a lower dimension. A supervised DNN is subsequently developed based on the lower dimensional data to characterize the phases and phase transitions. As a proof of concept, we employ this framework to study the coil to globule phase transition of a model polymer system. We conduct coarse-grained MD simulations to collect MD trajectories of a single polymer chain over a wide range of temperatures and use the dPOLY framework to predict polymer phases. The dPOLY framework accurately predicts the critical temperatures for the coil to globule transition for a wide range of polymer sizes. This method is generic and can be extended to capture various other phase transitions and dynamical crossovers in polymers and other soft materials. It can also significantly accelerate polymer phase prediction and characterization.



■ INTRODUCTION

Phase transitions in polymers are governed by correlated microscopic interactions and relaxation of their large number of atoms and segments over multiple time and length scales.[1−3] Given the wide variations in microscopic degrees of freedom and macroscopic properties exhibited by polymers, identifying new phases and phase transitions from molecular dynamics (MD) trajectories can be challenging. Phase transitions are traditionally characterized via an abrupt change in an order parameter representing a system's local structures. The order parameter is chosen such that it is sensitive to the specific character of a transformation and can provide insightful information about the transition. However, there is no universal choice of order parameters to capture the wide range of phase transformations in polymers and other soft materials. Also, there is no single straightforward approach to identify metastable phases and dynamical crossovers such as vitrification, jamming, gel formation, and localization transition.[3,4] Identifying such complex crossovers and phase transitions require a priori knowledge of the transformations. Furthermore, there are many materials science problems where no conventional order parameter exists,[5] especially disorder phase transitions such as liquid to liquid, liquid to amorphous solids, and metastable phases are hard to identify. Accurate detection and systematic characterization of phase transitions and dynamical crossovers often require simulations in a very specific ensemble and advanced sampling, which can be computationally expensive. Therefore, discovering new phase transitions and dynamical crossovers requires developing a generic strategy that does not need a priori information. To this front, machine learning (ML) methods are powerful in recognizing minute changes in a data set's trends and patterns. There are many recent emphases on developing simple yet generic ML-based strategies for identifying and quantifying phase transitions in materials systems.[6−10] Motivated by these studies, herein we aim to develop a generic ML framework for autonomous identification and characterization of phase transitions and dynamical crossovers in the MD trajectory of polymers.

There are two approaches for ML—unsupervised and supervised methods to derive useful inferences from data sets. In unsupervised ML, the learning algorithm receives unlabeled data, wherein the task is to extract features or make a subgroup of data based on specific characteristics (known as clustering analysis). On the other hand, during supervised ML, data are supplemented by labels that distinguish a subset of data from the rest of the data set. During supervised ML, the

Received: November 30, 2020
Revised: March 2, 2021
Published: March 16, 2021

learning algorithm learns the data's mapping to its corresponding labels and subsequently predicts the unknown data labels during the test cycle. Although many supervised and unsupervised ML methods are used to detect phases, phase transitions, and crossovers in condensed matter systems,[11,12] their adaptation in polymers and soft materials are not common. Here, we explicitly focused on a subclass of supervised and unsupervised ML tools, viz., deep learning for predicting phases in a polymer's MD trajectory. We introduce dPOLY, a deep learning framework that captures polymer conformations' variations over a thermophysical process such as cooling, compression, and drying and determines the phases and their crossover points. This framework consists of two major components:

i) an unsupervised deep autoencoder to map the trajectory to a lower dimension and
ii) a supervised deep neural network (DNN) that is built on the lower dimensional polymer conformation to predict the phases and phase transitions.

The unsupervised deep autoencoder reduces the number of features of a polymer structure by choosing the most important ones that still retain the essence of the polymer structure.[13,14] This reduction of dimensionality of a $3N$ dimensional polymer conformation reduces the number of input parameters of a predictive model and provides a better descriptor of a polymer conformation for phase prediction, eventually improving the DNN classification accuracy. We demonstrate this framework for a prototypical example of a coil to globule transition of a polymer chain undergoing cooling.

Chain conformation in dilute solution, notably the coil−globule transition (CGT), has long been studied due to its theoretical importance and diverse applications.[15] Molecular simulations—Monte Carlo and MD—with either implicit or explicit solvent models and mean-field level theories have been extensively used to understand the influence of chain length, chain stiffness, confinement, and other effects on the CGT transitions and their order and transition dynamics.[16−27] The CGT is typically determined by an abrupt change in the polymer's macroscopic properties, such as specific heat and radius of gyration. Accurate quantification of CGT requires advanced sampling methods such as integrated temperature sampling[16] and Wang−Landau algorithm.[23] Unlike these conventional approaches, here, our goal is to predict the phase transition by analyzing the standard MD simulation trajectory of a polymer without any advanced sampling to establish and validate a data-driven framework for identifying and characterizing phases and phase transitions.

Within the dPOLY framework, the data are generated by conducting MD simulations of a polymer chain in an implicit solvent condition by solving its Langevin equation of motion across a wide range of temperatures. An autoencoder neural network is developed using all the polymer conformations that are collected during the simulated annealing of a polymer. Subsequently, a feed-forward back-propagation neural network is developed based on the polymer's high-temperature and low-temperature conformations. The dPOLY framework accurately predicts the coil and globule phases of all the polymer conformations. It also provides an accurate estimation of CGT temperature for a wide range of polymer chain lengths. The crossover temperature exhibits a power law behavior with the polymer chain length, which agrees with the previous studies. Beyond its current application to the coil to globule transitions in a coarse-grained polymer model, the dPOLY framework is extensible for chemically realistic polymer models with wide molecular structure variations, chemistry, and phases. As in this application, this approach is likely to yield new physical understanding and capturing new phases, including metastable ones for polymers and other soft materials. Moreover, the dPOLY framework can work autonomously without much human intervention. It requires a relatively less amount of data than a pure molecular simulation-based prediction of phase transition and can accelerate polymer phase discovery.

MODELS AND METHODS

The dPOLY framework is developed while testing it to identify and characterize the coil to globule transition of a model polymer system. In this section, we first describe the polymer model and the individual components of the dPOLY framework. Subsequently, we construct the workflow of dPOLY that integrates MD simulation, unsupervised deep learning, and supervised deep learning methods.

Polymer Model. We use a generic coarse-grained model of homopolymers for validating the dPOLY framework in identifying the coil to globule transition. In this model system, two adjacent coarse-grained monomers of a polymer are connected by the finitely extensible nonlinear elastic (FENE) potential[28] of the form $V_{\text{FENE}} = -0.5KR_0^2\ln[1-(r/R_0)^2]$, where, $k = 30\epsilon/\sigma^2$ and $R_0 = 1.5\sigma$. Any two monomers in the system interact via the Lennard−Jones (LJ) potential of the form $V(r_{ij}) = 4\epsilon_{ij}[(\sigma/r_{ij})^{12} - (\sigma/r_{ij})^6]$. ϵ_{ij} is the interaction energy between any two monomers i and j. The size of all the monomers is σ. The LJ interaction is truncated and shifted to zero at a cut-off distance $r_c = 2.5\sigma$ to represent attractive interaction among the monomers.

MD Simulations and Data Generation. Implicit solvent MD of polymer chains are simulated for a range of temperatures that incorporate the coil to globule transition. The initial configuration of a polymer chain is placed in a cubic simulation box of fixed dimensions. The simulation box is periodic in all three dimensions. The number density of particles in the simulation box is 0.001. The equation of motion are integrated using the Velocity Verlet algorithm with a time step of 0.001τ, where $\tau = \sigma\sqrt{m/\varepsilon}$ is the unit of time. The temperature of the systems is controlled using the Langevin thermostat within the LAMMPS (Large-scale Atomic/Molecular Massively Parallel Simulator) environment.[29] The simulation is conducted for polymers of chain lengths N = 30, 50, 100, 200, 300, 400, 500, and 800. We generate polymer trajectories for a range of temperatures starting from a very high value and ending at a low value via successive cooling, equilibration, and production cycles. At each cycle, the temperature is decreased by $0.04T^*$. A total of 60 simulations across the temperature range are conducted for each chain length. For a given polymer chain, the temperature range is chosen in such a way that it captures the CGT. Here, the temperature $T^* = Tk_B/\epsilon$ is in reduced LJ unit, where k_B is the Boltzmann constant. For any given temperature, the system is equilibrated for 10^8 MD steps, followed by a production cycle. During the production cycles, polymer frames are stored in every 10^4 MD steps. Each frame represents $3N$ position coordinates of a polymer of chain length N. We collect 10^4 frames from two extreme temperatures and 10^3 frames of polymer conformations from each of the intermediate temperatures. A total of 78,000 frames are collected for each polymer chain length. All the data are used to train, validate, and test the autoencoder. We use 50 and 10% of the data to train and validate the autoencoder, respectively. The remaining data are used for testing the performance of the autoencoder model. For the DNN model, 20,000 data from the two extreme temperatures are used to train, validate, and test the model. The remaining data collected from the intermediate temperatures are used to identify the CGT. We note that all the data are randomized before the model development. These data are not time series data and they represent phase transformation as a function of temperature.

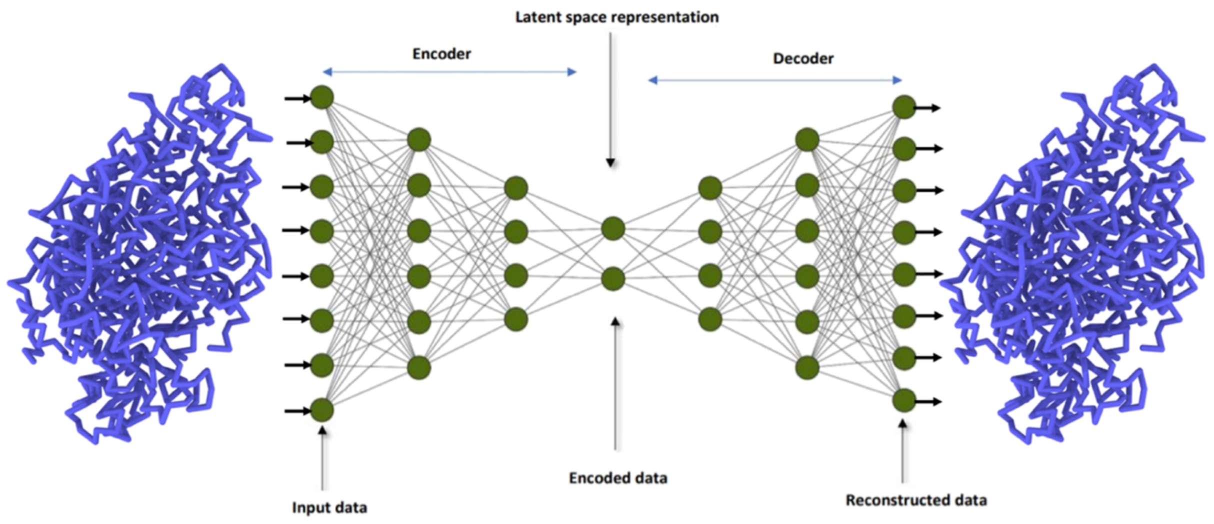


Figure 1. Autoencoder neural network. It receives the position coordinates of all the monomers of a polymer chain through its input layer neurons. The position coordinates are mapped to a lower dimensional space by the encoder part of the network. The lower dimensional representation is reconstructed to the actual higher dimension, viz., the position coordinates of the monomers of the polymer by the decoder part of the network. The number of nodes in the input layer and the output layer (reconstruction layer) are the same. The number of latent variables or the dimension of the feature space depends on the polymer chain length.

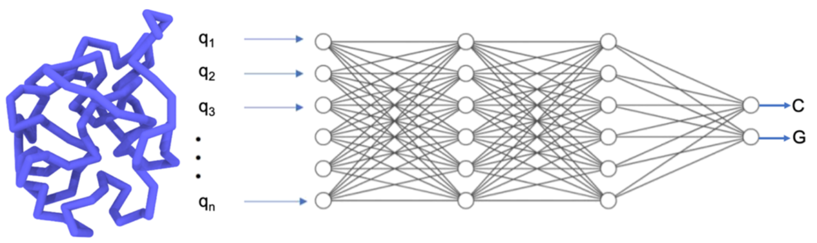


Figure 2. Supervised DNN. The position coordinates of all the monomers of a polymer conformation in its feature space serve as the input of the neural network. Only the labeled configurations are used for the training of the network. The trained network is used to predict labels of all the frames in a trajectory. The network has two outputs—C and G—that represent the labels, viz., state variables. Here, $C = 1$ and $G = 0$ represent the coil phase while $C = 0$ and $G = 1$ define the globule phase.

Unsupervised Deep Autoencoder. An autoencoder is an unsupervised learning method in which a neural network is used for the task of representation learning. A neural network architecture is designed in such a way that it compresses the correlations in an input data set to a lower dimensional representation and consequently decompresses them back to their original correlations. This compression and decompression of data and their correlations are achieved via a bottleneck neural network structure as shown in Figure 1. The autoencoder has two fragments—an encoder and a decoder. The encoder compresses the data into a lower dimensional latent space representation, also known as feature space. The decoder expands the lower dimensional representation into their actual higher dimensional representation. In the current study, the autoencoder compresses $3N$ positional coordinates of a polymer chain of length N into a lower dimensional latent space representation. The network contains an intermediate "bottleneck" layer that consists of fewer nodes than the input and output layers. This particular architecture forces the network to learn a compact representation of the input data, that is, the polymer coordinates. Therefore, it allows automatic discovery of polymer configuration features in low dimensional representation and helps classify the phases. Thus, this autoencoder helps visualize and characterize the collective behavior and correlations of a large number of interactive particles in a many-body system in a lower dimension, which is otherwise inconvenient to identify in its actual higher dimension.

The above task is achieved by a feed-forward back-propagation neural network with multiple intermediate layers, as shown schematically in Figure 1. An autoencoder of this topology is also known as a deep autoencoder due to the presence of multiple intermediate layers. Within an autoencoder topology, both the encoder and decoder segments have an equal number of neurons. Our network structure and the number of neurons in its layers can be written as $3N - n_1 - n_2 - n_3 - n_4 - q - n_4 - n_3 - n_2 - n_1 - 3N$, where each of these indices represents the number of neurons in the corresponding layers. Here, the number of neurons in the input and output layers are $3N$ for a polymer of chain length N. The input layer and the output layer are identical as they hold the same position coordinates of a polymer chain. n_1, n_2, n_3, and n_4 are the number of neurons in the four intermediate layers in-between the input layer and the middle layer. Similarly, n_4, n_3, n_2, and n_1 are the number of neurons in the four intermediate layers in-between the middle layer and the output layer. Therefore, the encoder and decoder part of the network are mirror images of each other. The middle layer has q neurons that represent the dimension of the feature space. n_1, n_2, n_3, n_4, and q are system-specific, and system-to-system adjustments are done to minimize the error in reproducing the exact structure of a polymer of chain length N during the autoencoder training.

Each neuron of a given layer is connected to all the adjacent layer neurons by weight parameters in a directed acyclic graph. A neuron receives the weighted sum of signals from all the neurons of its previous layer and activates it by an activation function[30] and then feeds it to all the neurons in its next layer. We use a rectified linear unit (ReLU) function[31] for activating signals in all the neurons in the network's intermediate layers. No computation is done at the input layer neurons that hold the input values. The output layer neuron activates the signal via a linear activation function and provides the output. The network is trained by a feed-forward back-propagation method. During the training, the weights between all the neurons'

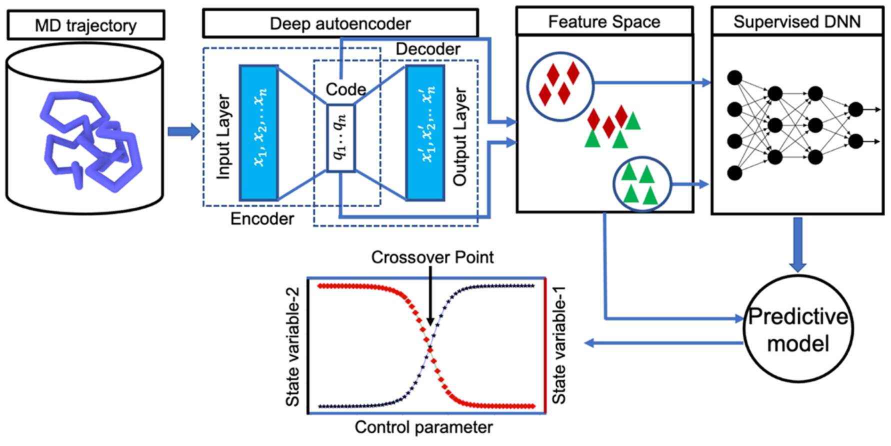


Figure 3. dPOLY workflow. The workflow consists of five sequential steps—(I) Generation of an MD trajectory using simulation. (II) All the frames of a trajectory are used to build an autoencoder. The trained autoencoder produces the code for all the frames in a feature space. (III) The frames that are far away from the crossover points are labeled by state variables in the feature space. (IV) The labeled frames are then used to develop a DNN predictive model. (V) The model predicts the labels of all the frames in the feature space, which are further analyzed for accurate identification of crossover points.

pairs are adjusted to minimize the error, measuring the difference between the predicted values and the expected reference values at the output layer neurons. The loss function of the network represents the error. We use Adam optimization,[32] a first-order gradient-based optimization method, with a learning rate of 0.001 to optimize the autoencoder's cost function. During the training period, the data are fed in batches to the network repeatedly until a termination criterion is achieved. Typically, the training continues until a reasonably low value of the loss function is achieved and it reaches a plateau. We stop the training when the loss function is considerably low and not improving over four to five cycles. More details of the training and development of a feed-forward back-propagation neural network can be found in our previous works.[33,34] Once the autoencoder is trained, it can be subjected to a polymer conformation for estimating its projection to a lower dimension. The trained autoencoder produces an equivalent polymer trajectory at the output layer and its corresponding encoded values in the middle layer's neurons. The encoded values derived from the middle layer's neurons represent the lower dimensional mapping of the polymer configurations, which are used for further analysis.

Supervised DNN. A separate DNN is established for the supervised learning of the polymer chain conformations labeled by their state variables. The supervised DNN aims to build a nonlinear heuristic model for the relationship between the input variables (e.g., polymer position coordinates) and output variables (e.g., coil state or globule state). A schematic of a supervised DNN for polymer phase prediction is shown in Figure 2; it consists of neurons organized into an input layer, two hidden layers, and one output layer. The input layer consists of nodes representing polymer coordinates in its feature space as determined by the autoencoder. The output layer contains two nodes representing a pair of state variables that define the phase of a polymer conformation. All the neurons in the intermediate layer activate the signal via the ReLU activation function, similar to the autoencoder. The output layer neurons activate the signal via a sigmoidal activation function[35] and produce the output. We note that a DNN with a large number of parameters often suffers from overfitting. We use a dropout regularization procedure[36] to prevent such overfitting. According to this procedure, the weights of a small fraction of intermediate layer neurons are updated in a given cycle. This prevents neurons from adapting too much. More details of the dropout mechanism can be found elsewhere.[36] We use a commonly used dropout rate within a range of 0.2−0.5. The network is trained by a feed-forward back-propagation method similar to the autoencoder training process. The optimal numbers of neurons in the hidden layers have no universal values but are selected in an effort to maximize the efficiency and accuracy of the DNN. Similar to the autoencoder, Adam optimizer[32] is used with a default learning rate of 0.001 to optimize the cost function of the DNN. The trained DNN can be used to predict the labels, that is, state variables of a polymer conformation.

Deep Learning of Polymers (dPOLY). The workflow of dPOLY is schematically shown in Figure 3, which consists of five sequential steps as discussed below.

1. We generate a polymer trajectory using MD simulation as described above. The trajectory consists of polymer conformations during the thermophysical process, such as cooling. During the cooling cycle, we collect equilibrium configurations of polymers for 60 different temperatures. At each temperature, 10,000 frames are collected.
2. An autoencoder model is built using all the configurations from the MD trajectory for dimensionality reduction and feature selection. The autoencoder is trained by all the frames of the polymer trajectory. Once the autoencoder is trained, it is subjected to all the frames of the same polymer trajectory that consists of configurations for the entire range of temperature. The network produces an equivalent polymer trajectory at the output layer. We extract the signal values from the intermediate "bottleneck" layer, representing the lower dimensional mapping of the polymer configurations within the temperature range for further study. Here, each data point in the lower dimensional map represents a single polymer configuration within the temperature range.
3. We introduce state variables to characterize phases of polymer conformations in the feature space distinctly. We only label the data that are far away from the crossover regions, typically conformations drawn from two extreme temperatures. For the case of one phase transformation, such as CGT, two state variables, C and G, are introduced. The coil state is represented by $C = 1$ and $G = 0$, and the globule state is represented by $C = 0$ and $G = 1$ for further ML purposes.
4. The labeled data of step 3 are considered as the representations of a polymer's pure phases in the MD trajectories. The polymer configurations that are labeled in step 3 are then used to develop the DNN. The labeled polymer configurations are randomly selected for the training and testing of the DNN. We use 50% of labeled data for training and 10% to validate the network. The training cycles continue until the errors in predicting the validation data set's state variables are below 1% of their actual values.
5. We use the trained DNN for predicting labels of all the configurations of the MD trajectories that are mapped to the feature space. We note that the DNN is trained with selected

configurations far away from the crossover points, but it is used to create labels of all the feature space configurations. The labels, that is, the state variables, are then plotted as a function of the controlling parameter to identify the crossover points. Here, the temperature is the controlling parameter for the present case study of CGT. For each temperature, the state variables of 1000 configurations are predicted by the DNN for averaging purposes. We plot average state variables as a function of temperature. The intersection of these curves is defined as the crossover point of the phase transformation.

The dPOLY framework, along with its associated autoencoder and DNN, is developed using the Keras deep learning application programming interface.[37] The dPOLY framework is built for a general purpose artificial intelligence (AI) tool to identify phases, phase transitions, and dynamical crossovers in MD trajectories of polymers undergoing thermophysical treatments such as cooling, drying, and compression.

RESULTS AND DISCUSSION

Unsupervised Learning of Polymer Structures. The polymer conformations during the cooling simulations are used to develop an unsupervised deep autoencoder. During the training, $3N$ position coordinates of a polymer chain that are part of the training data set are repeatedly fed to the network in batches via the autoencoder's input layer. The output layer neurons of the autoencoder hold the same value as the input layer neurons. The weights between any two neurons are optimized during the training cycle to minimize the network's loss function that determines the difference between the input layer and the output layer values of all the $3N$ position coordinates. The central problem in developing a DNN model is choosing the number of training epochs to use. Too many epochs can often lead to overfitting of the training data set, whereas very few epochs result in an underfit deep learning model. To overcome this problem, we use early stopping criteria to specify an arbitrarily large number of training epochs and stop training the model once its performance stops improving on the validation data set of polymer conformations. For all the cases, we run 20 training cycles to develop the autoencoder model, beyond which the model's performance does not improve. During the autoencoder training, the loss function is shown in Figure 4 for three representative cases for

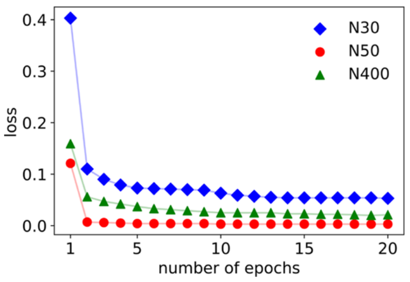


Figure 4. Loss functions of the autoencoder during its training for 20 epochs are shown for polymers of chain lengths N = 30, 50, and 400 for their respective validation data sets.

polymer chain lengths N = 30, 50, and 400. We define loss function as the mean-squared error in predicting monomer positions, which can be written as $L_{\text{autoencoder}} = \frac{1}{s}\sum_{i=1}^{s}(x_i - x_i')^2$. Here, s is the number of configurations in the validation data set. A position coordinate of a monomer in the input layer neuron and output layer neurons is represented as x_i and x_i', respectively. For all the case studies, the loss functions show plateau within the 20 training cycles. This indicates that the autoencoder has learned the polymer structures for the entire range of temperature very accurately. We note that eight autoencoders are developed in this study, each corresponding to a polymer of a specific chain length. As mentioned in the method section, the topology of all the autoencoders is $3N - n_1 - n_2 - n_3 - n_4 - q - n_4 - n_3 - n_2 - n_1 - 3N$, where N is the polymer chain length. The neurons in the intermediate layers n_1, n_2, n_3, and n_4 are varied from system to system in order to improve the accuracy. However, it is possible to fix the values of n_1, n_2, n_3, and n_4 across different chain lengths, which will result in a slight variation in loss functions without any significant changes in the featured space representation of the conformations. Similarly, the feature space dimension is also adjusted to improve the accuracy. For example, we identify $q = 6$ for $N = 30$ and $q = 150$ for $N = 800$ that yield the lowest error in the autoencoder's prediction. The exact details of all the autoencoders' topologies are summarized in the Supporting Information. The sensitivity of a loss function with the variation of neurons in the intermediate layers is also shown in the Supporting Information.

Now we test the autoencoder's performance in predicting the polymer structure for the test data set. The snapshot of a randomly chosen polymer conformation from the test data set is fed to the trained autoencoder's input layer, and the reconstructed polymer configuration's snapshot in its output layer is shown in Figure 1. It visually suggests that the trained autoencoder replicates a polymer configuration in its output layer with reasonable accuracy. Moreover, we calculate the radius of gyration of individual polymer frames that are passed to the autoencoder along with its predicted configurations for the test data set. Figure 5 shows the predicted radius of gyration of a polymer conformation as a function of its reference values for three representative cases of N = 50, 100, and 200. The mean absolute errors (MAEs) for all the cases are within a range of 0.02−0.07. Therefore, the predicted R_g values of the autoencoders are in good agreement with the reference values.

Supervised Learning of Polymer Phases. We prepare a subset of polymer configurations far away from the crossover region, which serves as the training, validation, and test data sets for developing a supervised DNN. This subset consists of 20,000 configurations of the polymer chains sampled from a cooling cycle's highest and lowest temperatures. A label—C = 1 and G = 0, is assigned to the configurations that correspond to the first phase (high-temperature configurations), and another label—C = 0 and G = 1—is assigned for the configurations that correspond to the second phase (low-temperature configurations). We use 50% of the data for training, 10% of the data for validation, and the remaining data for testing the performance of the DNN. Figure 6 shows the loss function during the training of the DNN for three representative cases, viz., N = 30, 50, and 400. We employ the concept of cross-entropy to define the loss function of the DNN, which is known as the cross-entropy loss function.[38] The cross-entropy loss function of the DNN can be computed as $L_{\text{DNN}} = -[t\cdot\ln(p) + (1 - t)\cdot\ln(1 - p)]$, where t is the true value of a label and p is its probability. The probability of a label can be computed as $p(t_i) = e^{t_i}/\sum_{i=1}^{2} e^{t_i}$. Here, t will have a value zero or 1 for this binary classification problem. It is to be noted that, unlike the state variables' binary value, the cross-

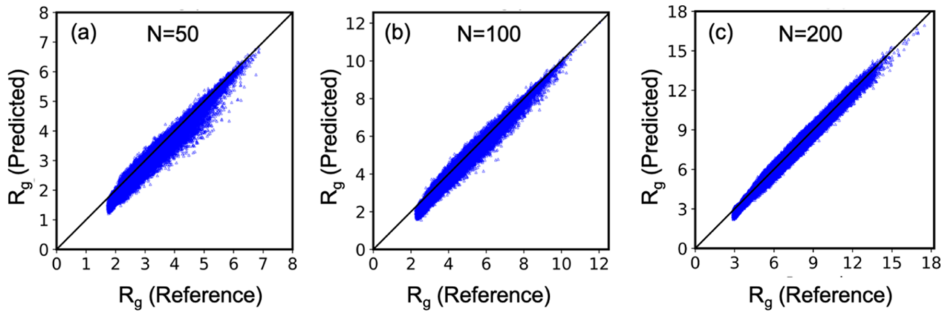


Figure 5. Performance of an autoencoder. The predicted radius of gyration is plotted against the reference values for chain lengths $N = 50$, 100, and 200 in (a−c), respectively, for the respective test data sets. The black lines represent the zero MAE in the predicted R_g value. The MAEs are 0.07, 0.04, and 0.02 for (a−c), respectively. The coefficients of determination (R^2) are 0.97, 0.98, and 0.99 for (a−c), respectively.

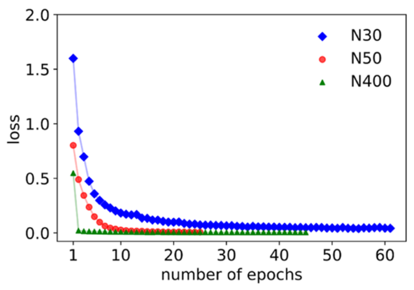


Figure 6. Loss function during the training cycle of the DNN model for three representative cases of $N = 30$, 50, and 400, respectively, for the validation data sets.

entropy function is continuous and differentiable. This particular approach makes it possible to calculate the loss function's derivative with respect to all the DNN topology weights, thus essential in identifying the network's optimal weights. During the training, the weights and biases of the DNN are updated by using gradient descent via the back-propagation algorithm to minimize the cost function. The smaller is the cross-entropy function, the better is the model. We have also adjusted the number of neurons of the intermediate layers of a DNN to incorporate the system-to-system variation. The exact topologies of the DNNs for different chain lengths are reported in the Supporting Information. A perfect DNN model has a cross-entropy loss of zero. Trainings are stopped when the loss function decreases and shows a plateau for both training and validation sets. The training cycles vary from 25 to 100, depending on the chain length, as shown in Figure 6. Within the training cycles, the loss function reaches a plateau near zero.

After the completion of training, we test the performance of the DNN for both known and unknown data, that is, for both training and test data sets. The predicted labels of the polymer structures are compared with the reference values for both the training and test sets in Figure 7 for a representative case of a polymer of chain length $N = 400$. For both the training and test sets, we split the data into two subsets, each for a reference phase. For example, the reference values of the labels are $C = 1$ and $G = 0$ for Figure 7a, and DNN predicted labels of all the configurations are $C \sim 1$ and $G \sim 0$. For all the cases of polymer chain lengths, the MAE of the DNN prediction is less than 1%. Therefore, we infer that the current DNN model very accurately learns the polymer structures' labels that represent two distinct phases of the system. This model is further used for predicting the phase transition of polymers during thermal annealing.

Predicting Phase Transition. An accurate prediction of phase transition temperature requires well-defined parameters representing a change in the state as a function of temperature. The DNN model can be used for this purpose. Here, we use the trained DNN model to predict the labels of all the polymer configurations across the entire range of temperatures. For each temperature, DNN is used to predict the labels for 1000 configurations. Figure 8 shows the average value of C and G as a function of temperature for three representative cases, $N =$

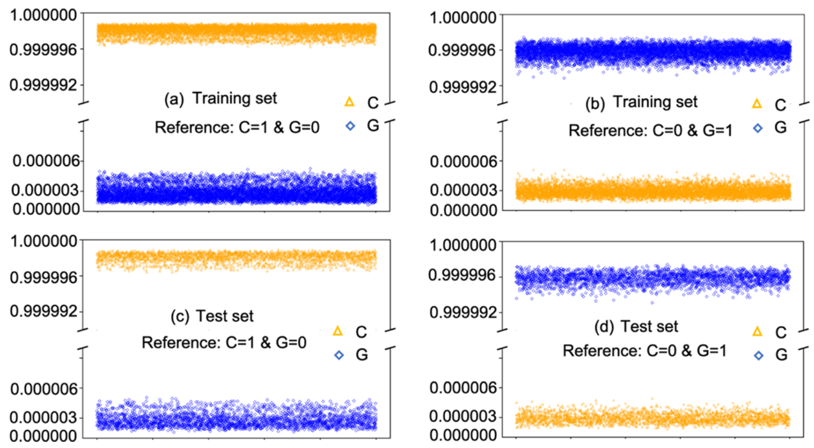


Figure 7. Performance of the DNN. The outputs of the DNN—C and G for training and test sets are shown in (a,b) and (c,d), respectively, for a representative case of $N = 400$. The references (labels) of conformations are as mentioned in the figures. The MAEs for all the cases are below 1%.

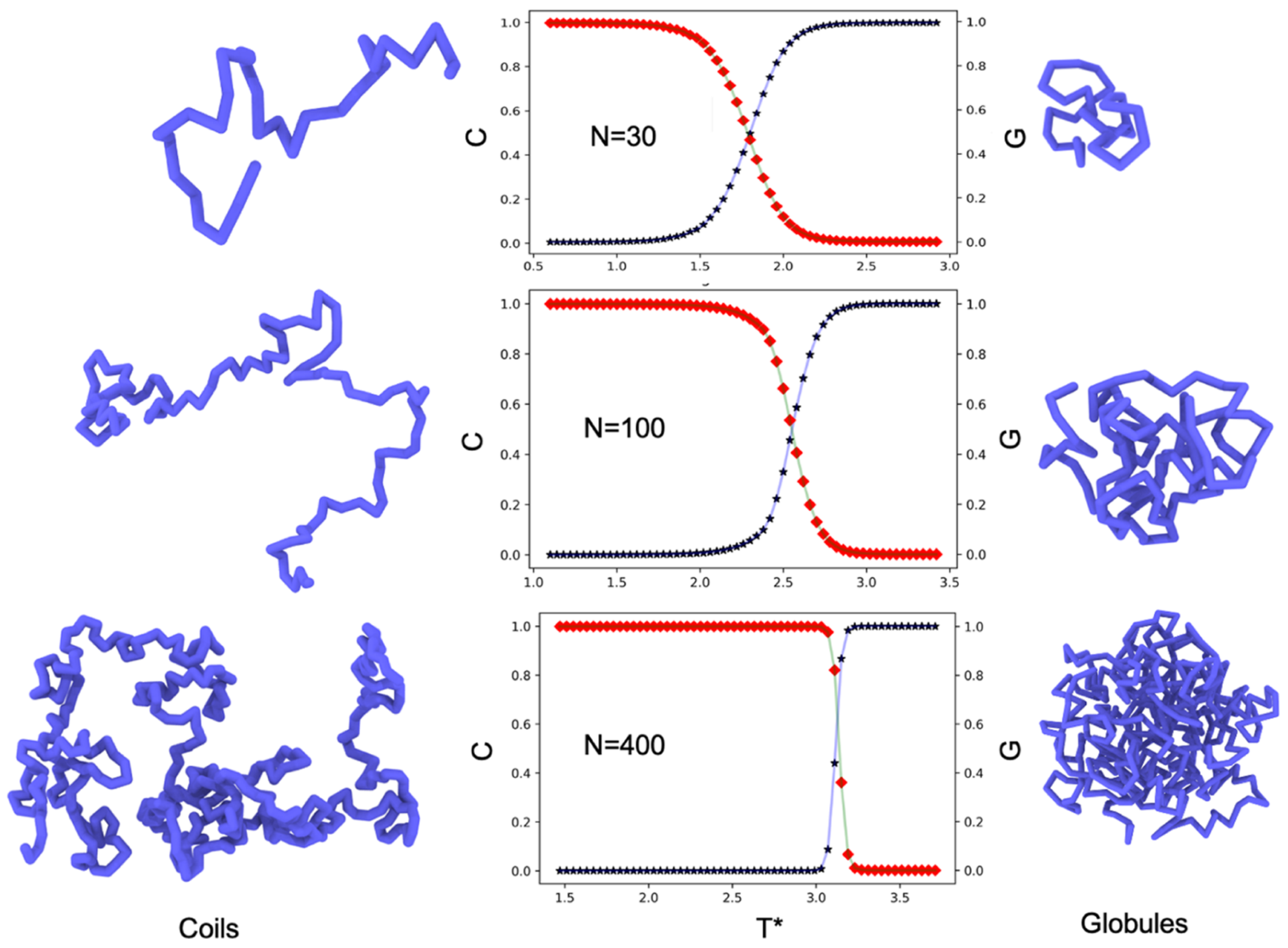


Figure 8. ML predicted coil to globule phase transition. The state variables C and G are plotted as a function of temperature for polymer chain lengths N = 30, 100, and 400. Left images correspond to C = 1 and G = 0. Right images represent C = 0 and G = 1.

30, 100, and 200. Although the DNN model is trained with the molecular configurations sampled from the two extreme temperatures, it predicts the extent of coil and globule phases of a polymer for all the intermediate temperatures. For all the cases, C and G show a sigmoid-type correlation as a function of temperature. At the low-temperature range, C remains close to zero, and it increases for an intermediate range of temperatures. For high temperature, C plateaus around 1. Similarly, G exhibits a value around 1 for the lower temperature range. It decreases in the intermediate range of temperatures and shows a plateau near zero for high temperature. We draw continuous curves for both C and G by fitting their respective data using a standard spline interpolation scheme. For all the cases, we observe that the C and G curves cross at an intermediate temperature. We infer this intermediate temperature where the C and G curves cross each other as the coil to globule transition temperature. The transition temperatures (T_c) for all the chain lengths are estimated from the crossover of the respective C and G curves and plotted as a function of $1/\sqrt{N}$ in Figure 9. The transition temperatures are in well agreement with the pure MD simulation-based prediction.[16] However, we note that the Wang–Landau algorithm suggests slightly lower values of the crossover temperatures.[23] In this work, we have generated the data using standard MD simulations without any advanced sampling techniques. Nevertheless, we expect that this MD–ML method's accuracy can be further improved by

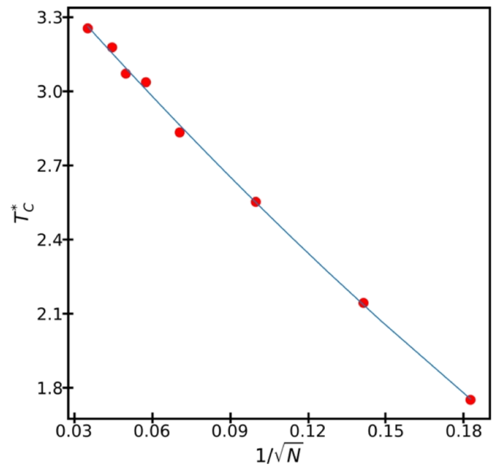


Figure 9. The Coil to globule crossover temperature is plotted as a function of $1/\sqrt{N}$, where N is the polymer chain length.

generating data with advanced sampling methods such as the Wang–Landau algorithm. The transition temperature increases with the chain length, and we fit $T_c(N) - T_\theta = a_1/\sqrt{N} + a_2/N$ to further understand the correlation. Here, a_1 and a_2 are the fitting parameters. T_θ is known as the theta temperature that characterizes coil to globule crossover of the model polymer. The solid line in Figure 9 represents the fitted line with $a_1 = -12.44$, $a_2 = 9.47$, and $T_\theta = 3.687$. It clearly suggests that the phase transition temperature exhibits a primary scaling dependence on the chain length N as $T_c \sim N^{-1/2}$. This prediction is well in agreement with the previously reported correlation using pure MD simulations with ITS.[16] This verifies that the dPOLY is an effective method for predicting phase transitions in polymers.

We note that a supervised DNN model can be developed based on the $3N$ position coordinates of a polymer structure without mapping it to a low-dimensional feature space via an autoencoder mapping. However, such a DNN based on $3N$ input layer neurons ($3N$-DNN) leads to larger errors in the predicted T_c values and the subsequent scaling relation, as shown in the Supporting Information. We find that an initial mapping of polymer confirmation to a lower dimensional representation helps improve the efficiency and predictability of a DNN classifier. Thus, the autoencoder is found to produce an efficient descriptor of a liner polymer chain for characterizing and predicting its phase behavior.

■ CONCLUSIONS

We report an order parameter-free approach to identify phases and phase transition in polymers. This approach is based on the deep learning of polymer conformation during a thermophysical process. The dPOLY framework consists of dimensionality reduction and classification of polymer structures across a controlling thermophysical parameter. Within the dPOLY workflow, an unsupervised autoencoder compresses the three-dimensional structure of a polymer chain to a lower dimensional latent space during encoding. The model successfully reconstructs the polymer conformation from the lower dimensional latent space representation to its actual three dimensions very accurately during decoding. This suggests that the model can generate a unique lower dimensional trajectory of the $3N$ dimensional MD trajectory. The lower dimensional data are then utilized to build a DNN classifier. The DNN classifier model predicts the state variables of all the polymer structures generated during a thermophysical process such as cooling or compression and identifying the phase of an unknown structure. The crossover of the state variables as a function of the controlling parameter is estimated as the phase transition's critical point. The advantage of dimensionality reduction (unsupervised learning) before building the DNN model is that it provides a better descriptor by reducing the number of input variables while retaining the data set's key features. Therefore, the DNN deals with a less number of input variables and, hence, performs better. Here, unsupervised learning is just a pre-processing of the data for developing a better predictive model. This two-step process is central to the dPOLY framework, wherein unsupervised learning reduces the number of input variables and supervised learning assigns labels to a polymer conformation. We employ the dPOLY method for predicting coil and globule phases of a model polymer chain undergoing thermal annealing. The dPOLY framework accurately predicts the coil to globule transition temperature for a wide range of polymer chain length. The direct use of monomer coordinates as an input into dPOLY underlies the robustness and simplicity of this approach. Another advantage of such an MD−ML framework is that it can accelerate the polymer phase transition prediction. A DNN model development needs molecular configurations only for the two extreme temperatures and a few configurations in the intermediate temperatures to identify the crossover temperature. For example, we have built the DNN model with 20,000 configurations sampled from the two extreme temperatures and use 1000 configurations from each of the intermediate temperatures. We have generated data for 60 temperatures for each polymer chain. In this way, we use a total of around 78,000 configurations to characterize and predict a phase transition of a polymer with a fixed chain length. However, a pure MD-based analysis and characterization of phases for 60 temperatures may require 60×10^5 configurations, assuming that 10^5 configurations are required for ensemble averaging of a macroscopic property, which is typical. Since the computing time to generate data grows linearly with the amount of data points, a pure MD-based characterization of phase transformation will be computationally much more expensive than the current ML-based method. We estimate that the current dPOLY method is over 50 times faster than a pure MD route of characterizing phase transformation. We further note that the present work did not aim for developing an ML model with a minimal amount of data; instead, it focused on the proof of concept of an ML-based characterization of polymer phase transformation. Therefore, it is quite possible to build ML models with even lower amounts of data, especially advanced concepts such as active learning to build ML models with sparse data.[39−41] Our future works will focus on building an ML-based polymer phase characterization technique with a minimal amount of data and a holistic comparison of the MD−ML approach with a pure MD route across diverse polymeric systems, including chemically realistic models. Although dPOLY is tested for the coil to globule transition, we expect this approach to identify other phase transitions and dynamical crossovers. For instance, dPOLY can be extended for systems with more than two phases and multiple controlling parameters. Besides, this deep learning framework does not require a priori knowledge of phase transitions and crossovers. Thus, it will be useful for capturing and characterizing complex processes such as glass formation and macromolecular crowding and discovering unknown phases and phase transitions in materials systems. We expect that the present work will engender the development of simple, generic, and robust AI tools suitable for macromolecular systems to characterize their structures, dynamics, and phase behavior.

■ ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acs.macromol.0c02655.

Additional details of the models include the topologies of the neural network and the autoencoder and T_c for all the chain lengths and comparisons between the performance of dPOLY and $3N$-DNN (PDF)

■ AUTHOR INFORMATION

Corresponding Author

Tarak K. Patra − *Department of Chemical Engineering, Indian Institute of Technology Madras, Chennai, Tamil Nadu 600036, India*; orcid.org/0000-0002-6002-0922; Email: tpatra@iitm.ac.in

Author

Debjyoti Bhattacharya – *Department of Chemical Engineering, Indian Institute of Technology Madras, Chennai, Tamil Nadu 600036, India*

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.macromol.0c02655

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

The work is made possible by the financial support from SERB, DST, Govt of India, through a start-up research grant. T.K.P. acknowledges the ICSR, IIT Madras, for the initiation, seed, and exploratory research grants. This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under contract no. DE-AC02-06CH11357. We also used the computational facility of the Center for Nanoscience Materials. Use of the Center for Nanoscale Materials, an Office of Science user facility, was supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, under contract no. DE-AC02-06CH11357. We acknowledge the use of the computing resources at HPCE, IIT Madras.

■ REFERENCES

(1) Schneider, H. A. Flexibility and Phase Transitions of Polymers. *J. Appl. Polym. Sci.* **2003**, *88*, 1590−1599.
(2) Egorov, V. M.; Yakushev, P. N. Phase and Relaxation Transitions in Poly(Tetrafluoroetylene). *Phys. Solid State* **2018**, *60*, 1874−1878.
(3) Cheng, S. Z. D. *Phase Transitions in Polymers: The Role of Metastable States*, 1st ed.; Elsevier Science: Amsterdam, Boston, 2008.
(4) Hung, J.-H.; Patra, T. K.; Meenakshisundaram, V.; Mangalara, J. H.; Simmons, D. S. Universal Localization Transition Accompanying Glass Formation: Insights from Efficient Molecular Dynamics Simulations of Diverse Supercooled Liquids. *Soft Matter* **2019**, *15*, 1223−1242.
(5) Carrasquilla, J.; Melko, R. G. Machine Learning Phases of Matter. *Nat. Phys.* **2017**, *13*, 431−434.
(6) Jadrich, R. B.; Lindquist, B. A.; Truskett, T. M. Unsupervised Machine Learning for Detection of Phase Transitions in Off-Lattice Systems. I. Foundations. *J. Chem. Phys.* **2018**, *149*, 194109.
(7) Jadrich, R. B.; Lindquist, B. A.; Piñeros, W. D.; Banerjee, D.; Truskett, T. M. Unsupervised Machine Learning for Detection of Phase Transitions in Off-Lattice Systems. II. Applications. *J. Chem. Phys.* **2018**, *149*, 194110.
(8) van Nieuwenburg, E. P. L.; Liu, Y.-H.; Huber, S. D. Learning Phase Transitions by Confusion. *Nat. Phys.* **2017**, *13*, 435−439.
(9) Wei, Q.; Melko, R. G.; Chen, J. Z. Y. Identifying Polymer States by Machine Learning. *Phys. Rev. E* **2017**, *95*, 032504.
(10) Vandans, O.; Yang, K.; Wu, Z.; Dai, L. Identifying Knot Types of Polymer Conformations by Machine Learning. *Phys. Rev. E* **2020**, *101*, 022502.
(11) Hu, W.; Singh, R. R. P.; Scalettar, R. T. Discovering Phases, Phase Transitions, and Crossovers through Unsupervised Machine Learning: A Critical Examination. *Phys. Rev. E* **2017**, *95*, 062122.
(12) Wang, L. Discovering Phase Transitions with Unsupervised Learning. *Phys. Rev. B* **2016**, *94*, 195105.
(13) Witten, I. H.; Frank, E.; Hall, M. A.; Pal, C. J. *Data Mining: Practical Machine Learning Tools and Techniques*, 4th ed.; Morgan Kaufmann: Amsterdam, 2016.
(14) Duda, R. O.; Hart, P. E.; Stork, D. G. *Pattern Classification*, 2nd ed.; Wiley-Interscience: New York, 2000.
(15) Nishio, I.; Sun, S.-T.; Swislow, G.; Tanaka, T. First Observation of the Coil−Globule Transition in a Single Polymer Chain. *Nature* **1979**, *281*, 208−209.
(16) Wang, W.; Zhao, P.; Yang, X.; Lu, Z.-Y. Coil-to-Globule Transitions of Homopolymers and Multiblock Copolymers. *J. Chem. Phys.* **2014**, *141*, 244907.
(17) Guo, J.; Liang, H.; Wang, Z.-G. Coil-to-Globule Transition by Dissipative Particle Dynamics Simulation. *J. Chem. Phys.* **2011**, *134*, 244904.
(18) Simmons, D. S.; Sanchez, I. C. A Model for a Thermally Induced Polymer Coil-to-Globule Transition. *Macromolecules* **2008**, *41*, 5885−5889.
(19) Zhang, X.-k.; Su, J.-y. Monte Carlo Simulation of Coil-to-Globule Transition of Compact Polymer Chains: Role of Monomer Interacting. *Chin. J. Chem. Phys.* **2018**, *31*, 784−788.
(20) Min, S. H.; Kwak, S. K.; Kim, B.-S. Atomistic Simulation for Coil-to-Globule Transition of Poly(2-Dimethylaminoethyl Methacrylate). *Soft Matter* **2015**, *11*, 2423−2433.
(21) Sotta, P.; Lesne, A.; Victor, J. M. The Coil−Globule Transition for a Polymer Chain Confined in a Tube: A Monte Carlo Simulation. *J. Chem. Phys.* **2000**, *113*, 6966−6973.
(22) Reddy, G.; Yethiraj, A. Implicit and Explicit Solvent Models for the Simulation of Dilute Polymer Solutions. *Macromolecules* **2006**, *39*, 8536−8542.
(23) Seaton, D. T.; Wüst, T.; Landau, D. P. Collapse Transitions in a Flexible Homopolymer Chain: Application of the Wang-Landau Algorithm. *Phys. Rev. E: Stat., Nonlinear, Soft Matter Phys.* **2010**, *81*, 011802.
(24) Yang, X.; Lu, Z.-Y. Control Globular Structure Formation of a Copolymer Chain through Inverse Design. *J. Chem. Phys.* **2016**, *144*, 224902.
(25) Vogel, T.; Bachmann, M.; Janke, W. Freezing and Collapse of Flexible Polymers on Regular Lattices in Three Dimensions. *Phys. Rev. E: Stat., Nonlinear, Soft Matter Phys.* **2007**, *76*, 061803.
(26) Kamata, K.; Araki, T.; Tanaka, H. Hydrodynamic Selection of the Kinetic Pathway of a Polymer Coil-Globule Transition. *Phys. Rev. Lett.* **2009**, *102*, 108303.
(27) Bejagam, K. K.; An, Y.; Singh, S.; Deshmukh, S. A. Machine-Learning Enabled New Insights into the Coil-to-Globule Transition of Thermosensitive Polymers Using a Coarse-Grained Model. *J. Phys. Chem. Lett.* **2018**, *9*, 6480−6488.
(28) Kremer, K.; Grest, G. S. Molecular Dynamics (MD) Simulations for Polymers. *J. Phys. Condens. Matter* **1990**, *2*, SA295−SA298.
(29) Plimpton, S. Fast Parallel Algorithms for Short-Range Molecular Dynamics. *J. Comput. Phys.* **1995**, *117*, 1−19.
(30) Hahnloser, R. H. R.; Sarpeshkar, R.; Mahowald, M. A.; Douglas, R. J.; Seung, H. S. Digital Selection and Analogue Amplification Coexist in a Cortex-Inspired Silicon Circuit. *Nature* **2000**, *405*, 947−951.
(31) Ramachandran, P.; Zoph, B.; Le, Q. V. Searching for Activation Functions. **2017**, arXiv:1710.05941. arXiv preprint.
(32) Kingma, D. P.; Ba, J. Adam: A Method for Stochastic Optimization. **2017**, arXiv:1412.6980. arXiv preprint.
(33) Patra, T. K.; Meenakshisundaram, V.; Hung, J.-H.; Simmons, D. S. Neural-Network-Biased Genetic Algorithms for Materials Design: Evolutionary Algorithms That Learn. *ACS Comb. Sci.* **2017**, *19*, 96−107.
(34) Patra, T. K.; Loeffler, T. D.; Chan, H.; Cherukara, M. J.; Narayanan, B.; Sankaranarayanan, S. K. R. S. A Coarse-Grained Deep Neural Network Model for Liquid Water. *Appl. Phys. Lett.* **2019**, *115*, 193101.
(35) Torremolinos, S. From Natural to Artificial Neural Computation: International Workshop on Artificial Neural Networks, Malaga-Torremolinos, Spain, June 7−9, 1995: Proceedings. *International Workshop on Artificial Neural Networks 1995*; Springer-Verlag: Berlin; New York, 1995.
(36) Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929−1958.
(37) Keras: the Python deep learning API. https://keras.io.web (Accessed 2020-10-06).

(38) Rubinstein, R. Y.; Kroese, D. P. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning; Information Science and Statistics*; Springer-Verlag: New York, 2004.
(39) Loeffler, T. D.; Manna, S.; Patra, T. K.; Chan, H.; Narayanan, B.; Sankaranarayanan, S. Active Learning A Neural Network Model For Gold Clusters & Bulk From Sparse First Principles Training Data. *ChemCatChem* **2020**, *12*, 4796−4806.
(40) Loeffler, T. D.; Patra, T. K.; Chan, H.; Cherukara, M.; Sankaranarayanan, S. K. R. S. Active Learning the Potential Energy Landscape for Water Clusters from Sparse Training Data. *J. Phys. Chem. C* **2020**, *124*, 4907−4916.
(41) Loeffler, T. D.; Patra, T. K.; Chan, H.; Sankaranarayanan, S. K. R. S. Active Learning a Coarse-Grained Neural Network Model for Bulk Water from Sparse Training Data. *Mol. Syst. Des. Eng.* **2020**, *5*, 902−910.