Studying polymer physics by machine learning

by

Qianshi Wei

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Physics

Waterloo, Ontario, Canada, 2020

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Dr. Rene Fournier Professor Dept. of Chemistry, York University
Supervisor(s):	Dr. Jeff Z.Y. Chen Professor Dept. of Physics, University of Waterloo
Internal Member:	Dr. Russell Thompson Associate Professor Dept. of Physics, University of Waterloo
Internal Member:	Dr. Mark W. Matsen Professor Dept. of Chemical Engineering, University of Waterloo
Internal-External Member:	Dr. Nasser M. Abukhdeir Associate Professor Dept. of Chemical Engineering, University of Waterloo

Author's Declaration

Statement of Contributions

This thesis has 6 chapters and is written based on three co-author papers.

1. Results in Chapter 3: **Qianshi Wei**, Roger G Melko, Jeff Z. Y. Chen, *Phys. Rev. E*, **95**, 032504 (2017);
   I am the first author.

2. Results in Chapter 4: Xin Xu, **Qianshi Wei**, Huaping Li, Yuguo Chen, Yuzhang Wang, Ying Jiang, *Phys. Rev. E*, **99**, 043307 (2019);
   I am the co-first author with Miss Xu. We contributed equally to this work. I contributed to the researches on PCA. Miss Xu contributed to the researches on diffusion map. We both contributed to the researches on the hybrid approach.

3. Results in Chapter 5: **Qianshi Wei**, Ying Jiang, Jeff Z. Y. Chen, *Phys. Rev. E*, **98**, 053304 (2018);
   I am the first author.

As the first author, my responsibilities are conducting research, collecting results and drafting paper manuscripts. These researches are conducted under the supervision of Dr. Jeff. Z. Y. Chen, my supervisor.

Abstract

Recently, machine learning becomes a computational method that burst in popularity. Many disciplines, such as condensed matter physics, quantum chemistry, chemical engineering as well as polymer physics have incorporate machine learning into their studies. This thesis mainly focuses on applying machine learning methods into the study of polymer physics. More specifically, two computational methods are studied: 1. how to classify polymer states by supervised or unsupervised learning methods, 2. how to use FNNs to search for structures of diblock copolymer under self-consistent field theory scheme.

In the first topic, polymer samples that consist of both vastly different structures, such as gas-like random coil, liquid-like globular and subtly different structures, such as crystalline anti-Mackay, Mackay, are generated by Monte Carlo method. We then explored the capability of a FNN on the classification of different polymer configurations systematically. Base on a series of numerical experiments, we find that a FNN, after appropriate training, is able to not only identify all these structures, but also accurately locate the transition points between multiple states. The location given by the FNN has a good agreement with that provided by specific-heat calculations from the traditional method, which shows that the FNN offers a new tool for further studies of the polymeric phase transitions. We also studied these states with principal component analysis (PCA). When polymer samples only contain coil and globular states, PCA can distinguish these states, and offer insights to understand the relation between features and order parameters of these states. However, PCA itself is not powerful enough to distinguish globular, anti-Mackay, Mackay states. Then, a hybrid scheme combining PCA and supervised learning is utilized to identify and precisely detect the critical point of phase transitions between these polymer configurations.

Compared with traditional methods, our studies demonstrate machine learning based methods have some distinct advantages. Firstly, these methods directly and only use molecular coordinates, which indicates its high compatibility with multiple sampling methods. In addition, the trained FNN has high transferability. In terms of identify transition points, our approaches requires much fewer samples, which indicates they are computationally faster than the traditional methods.

In the second topic, we start from using the universal approximation theorem of FNN to build a machine learning based PDE solver. Our work mainly focuses on diffusion equations. This algorithm utilizes the function generated by the FNN as a trial function and adjusts the weights and biases of the FNN to search for the solution of a given PDE. The trial function will have a good match with the solution, when the weights and biases are optimal. Our approach is important to high dimensional diffusion equations. We discovered that the growth of the computational time obeys a power law with respect to the dimensionality, which indicates that the machine learning based solver offers a candidate algorithm that may not suffer from the “curse of dimensionality”.

We then demonstrated that this machine learning PDE solver can be conveniently adopted to deal with multi-variable, coupled integrodifferential equations in the self-consistent field theory for predicting polymer self-assembly structures. We observed all known three-dimensional classical structures, and our solutions have an excellent agreement with traditional solutions.

Acknowledgements

First of all, I'd like to express my appreciation to Prof. Jeff Z. Y. Chen for leading me into the researches of polymer physics, machine learning, and providing supervision throughout my Ph.D. studies. I also want to thank my advisory committee members composed of Prof. Rene Fournier, Prof. Nasser Abukhdeir, Prof. Mark Matson, and Prof. Russell Thompson, who provided many insightful suggestions on my projects and thesis. I also would like to thank Prof. Ying Jiang and his team for enormous help, collaboration and hospitality during my visit to Beihang university.

I am deeply indebted and grateful to my parents, in-laws and my beloved wife, Xiaodong. I can not reach this far without their endless supports and encouragements. I am also grateful to tons of helps from my friends and colleagues in Univeristy of Waterloo.

Lastly, I'd like to thank the financial support form Natural Sciences and Engineering Research Council of Canada, Beihang university and the computational resource from Compute Canada.

Dedication

To my family.

Table of Contents

List of Tables

List of Figures

Nomenclature

Abbreviations

BOO	Bond orientational order
FENE	Finitely extensible nonlinear elastic model
FNN	Feed-forward neural network
LSTM	Long short term memory
PCA	Principal component analysis
PDE	Partial differential equation
SCFT	Self-consistent field theory
WLC	Worm-like chain

Symbols

α_i	Penalty coefficient of machine learning based PDE solver
βi	Penalty coefficient of machine learning based PDE solver
χ	Flory-Huggins parameter
ϵ	Potential-well depth
γ	Specific-heat-like function
γ_i	Penalty coefficient of machine learning based PDE solver
ν	Label

ϕ_A	Density function of type A monomer
ϕ_B	Density function of type B monomer
σ	Activation function
τ	Error tolerance
C	Heat capacity
D	Dimensionality of input data
D'	Dimensionality of the data in low dimensional representation
E	Energy
e	Energy per monomer
F	Free energy
f	Volume fraction of type A monomer in a diblock copolymer
H	Hamiltonian
J	Cost function
k_B	Boltzmann constant
N	Polymerization
Q_l	Bond orientational order parameters
R_g	Radius of gyration
R_g	Radius of gyration
T	Temperature
T_C	Transition point
U	Interaction potential
W_A	External field acting on type A monomer
W_B	External field acting on type B monomer

Y_{lm}	Spherical harmonics of degree l, order m
$\mathbf{q}$	Output of an FNN

Chapter 1

Introduction

1.1 Motivation

A polymer is a macromolecule that consists of repeating units, called monomers, connected sequentially through covalent bonds. Polymers are significant for many reasons. To begin with, polymers are all around us, they are the building blocks of life. For example, DNA is composed of two strands of nucleic acids sequences that form a double helix structure. It carries genetic information and directs the manufacture of proteins. Proteins are also polymers composed of amino acids that have versatile functionality. Some proteins function as enzymes in cells that catalyze biological chemical reactions. Some proteins bind in membranes and behave as ion transporters that pump ions across the membrane to maintain a concentration gradient. Some proteins behave as signal transmitters between cells from different tissues. Beyond the existence in the organism, polymers are the component of many materials, such as cotton, rubber, plastic, glass, etc. Some of them are naturally existing, while some are artificial. Rubber, which is formed by polymers crossed-linked together, could be both natural or synthetic and has been used to make shoes, tires due to its elasticity. Polyethylene is the most common synthetic plastic materials, that is used as raw material for the manufacture of bags, containers. Acrylic and polycarbonate are good candidates for windshields on auto-motors.

Polymer chains have enormous variance in architectures. Fig. 1.1 demonstrates some typical examples. These architectures can be categorized based on monomer constituents and chain topology. Based on how many types of monomers exist along the chain, polymer chains can be categorized into two classes: homopolymer and heteropolymer. A homopolymer is a polymer chain that is composed of one single type of monomer, such

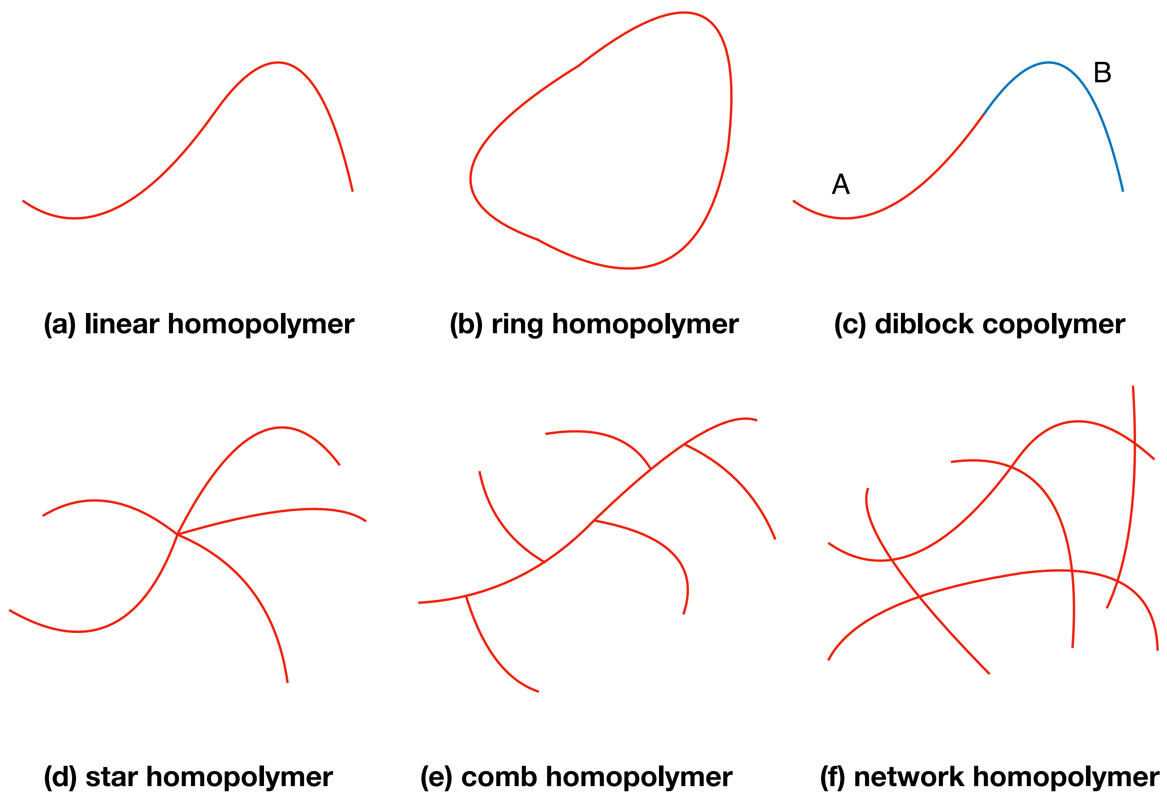

Figure 1.1: Sketches of some typical polymer architectures. Monomer type A are colored as red, B are colored as blue.

as a polypropylene chain which is composed of ethylene monomers. A heteropolymer is a polymer chain that contains more than one kind of monomers. If a heteropolymer is composed of multiple different homopolymers linked together, then it is called block copolymer. The simplest and extensively studied block copolymers are AB diblock copolymers, as sketched in Fig. 1.1(c). It is formed by connecting two homopolymers, one contains type A monomer, such as polystyrene molecules, another contains type B monomer, such as polyisoprene molecules, together like A-A-A-A-A-A-B-B-B. The number of type A monomers over the total number of monomers is denoted as f. Based on the topology, polymer chains can be linear, branched, cross-linked, or many other typologies. A polymer chain is "linear" if the sequential connection is single stranded. The simplest architecture is linear homopolymer as demonstrated in Fig. 1.1(a). When the two ends of a linear homopolymer are connected, it forms a ring homopolymer as shown in Fig. 1.1(b). Polymers can also be branched, that is, the sequential connection has multiple strands. Two examples are shown in Fig. 1.1(d)-(e). Fig. 1.1(d) is a sketch of the simplest branched polymers, star homopolymer, that consists of multiple linear homopolymers that linked together to a core, where the core could be as big as a molecule that have a large number of atoms or as small as a single atom. Fig. 1.1(e) is a sketch of a comb homopolymer which consists of some linear homopolymers grafted on a main chain. Multiple homopolymers can be cross-linked to form a network homopolymer, as shown in Fig. 1.1(f). Here the cross-link could be ionic or covalent bond formed between different homopolymers.

These architectures lead to a wide range of unique properties. Compared with simple fluids, such as water, some polymer fluids, such as asphalt, demonstrate high viscosity. Cross-linked polymers have elasticity different from traditional materials, such as rubbers which can be stretched to multiple time longer than its original length. Acrylic and polycarbonate have distinct transparency and brittleness compared with traditional glasses. We have seen that these structural properties of polymers can facilitate the design of soft functional or pharmaceutical materials, and the development of nanotechnology applications, etc [4, 6, 39, 116, 149, 161]. The studies of these properties not only stimulate the economic growth in the industrial markets, but also promote the discovery and understanding of novel behaviours in polymeric systems.

Polymer physics studies the phenomena due to chain connectivity. This field was pioneered by Hermann Staudinger, who experimentally justified the covalent bond connectivity [159]. Other pioneers, such as Werner Kuhn, Paul J. Flory, Prince E. Rouse Jr. and Bruno H Zimm, established the basic concepts of polymer physics [46]. Their works illustrate various static and dynamics properties [138, 197]. The development of experimental tools, such as neutron diffraction [27, 86] and light scattering [36], allowed the measurements of polymer conformations. These methods observed that polymer dissolved in solvent can be in different states based on the temperature, such as an expanded coil or collapsed globule [164]. Then, theoretical and numerical methods, such as molecular dynamics, Monte Carlo methods and field-based methods, in traditional statistical physics are applied to the studies of polymer physics. These theoretical methods allowed the research of polymer morphology, such as the dynamic of polymer structures under different temperatures [80]; searching, identifying new polymer states and linking the statistics of a polymeric system to phase transitions [34]. These methods can be broadly categorized into "particle-based simulation methods" or "field-based simulation methods". In particle-based simulation methods, the Hamiltonian we consider is composed of the kinetic energy and interaction potential which are formulated according to the monomer positions and momentums. It is intuitively understandable and straightforwardly implementable in the practical applications to the polymeric systems proposed under the coarse-grained scheme. There is an alternative approach, normally called field-based simulation, which imagines the polymers living in a continuous external field, $W(\mathbf{r})$, physically related to the spatially-varying chemical potentials, rather than the direct consideration of phase space of particle coordinates [48]. All these theoretical methods, along with experimental methods, lead to the discovery of a large number of polymer states.

1.2 Polymer states

Polymer state is one of the most important concepts in the study of polymer chain. Many physical properties, such as rigidity, viscosity, features, such as radius of gyration, knots, are strongly determined by its states. In this section, we will introduce two polymeric systems of interests, i.e. homopolymer and diblock copolymer, and the classical states obtained from simulation.

In modern simulations, a widely adopted way to simulate polymer chain systems is the coarse-grained approach. In this approach, a number of consecutive monomers are grouped together and viewed as one larger monomer. As an example, Fig. 1.2 shows a coarse-grained polypropylene chain. The interactions between monomers in coarse-grained models are modelled as bonded and non-boned. The interaction connecting two consecutive monomers along the backbone is called bonded, E_{bond}. The pair interaction between two non consecutive monomers is called non-bonded, E_{int}. The Hamiltonian of a coarse-grained polymer chain can be written as the summation of bonded and non-bonded interaction, as:

$$H(\mathbf{r}) = E_{\text{bond}} + E_{\text{int}}, \tag{1.1}$$

where $\mathbf{r} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$ is a chain configuration, $\mathbf{x}_i$ is the coordinates of ith monomer, and N is the number of monomers, also called polymerization. In general, the overall effects of these interactions will be repulsive when the distance between two monomers are small and attractive when the monomers are far away. Therefore, these interactions are trying to maintain an “optimal” distance between monomers.

Statistically, the equilibrium state of a polymeric system is determined by the minimum of the system free energy:

$$F = -TS + \langle H \rangle \tag{1.2}$$

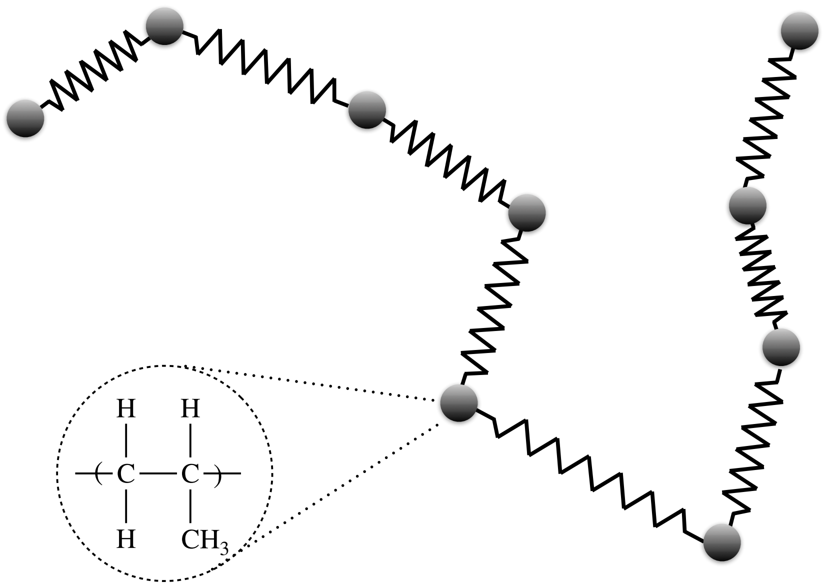

Figure 1.2: A graphical illustration of the coarse-grained model of a polypropylene chain.

where T is the temperature of the system. This free energy has an entropy part, S, which is determined by the total number of possible states, and an enthalpy part, $\langle H \rangle$, which is determined by the weighted average of the energy of different polymer chain configurations. The competition between the entropy and enthalpy leads to rich and complex states observed in polymeric systems [7, 130, 145, 146, 178, 185, 191]. Take a homopolymer solution as an example, Fig. 1.3 shows the states sampled from Monte Carlo methods. The algorithms and technical details are documented in Appendix. A. The polymer state in solution is determined by the "quality" of the solvent. Typically, when temperature is high, entropy dominates the system energy, which leads to bonds orienting randomly to occupy as many states as possible. As a result, the homopolymer will be in a random coil state. In this case, the solvent is called good solvent. Upon on the decrease of temperature, enthalpy will eventually outweigh entropy, and the homopolymer chain will collapse to a closed pack globule state [162, 188, 56, 177]. In this case, the solvent is called bad or poor solvent. Further reducing the temperature of the homopolymer chain may result in solid-solid transitions [145, 146].

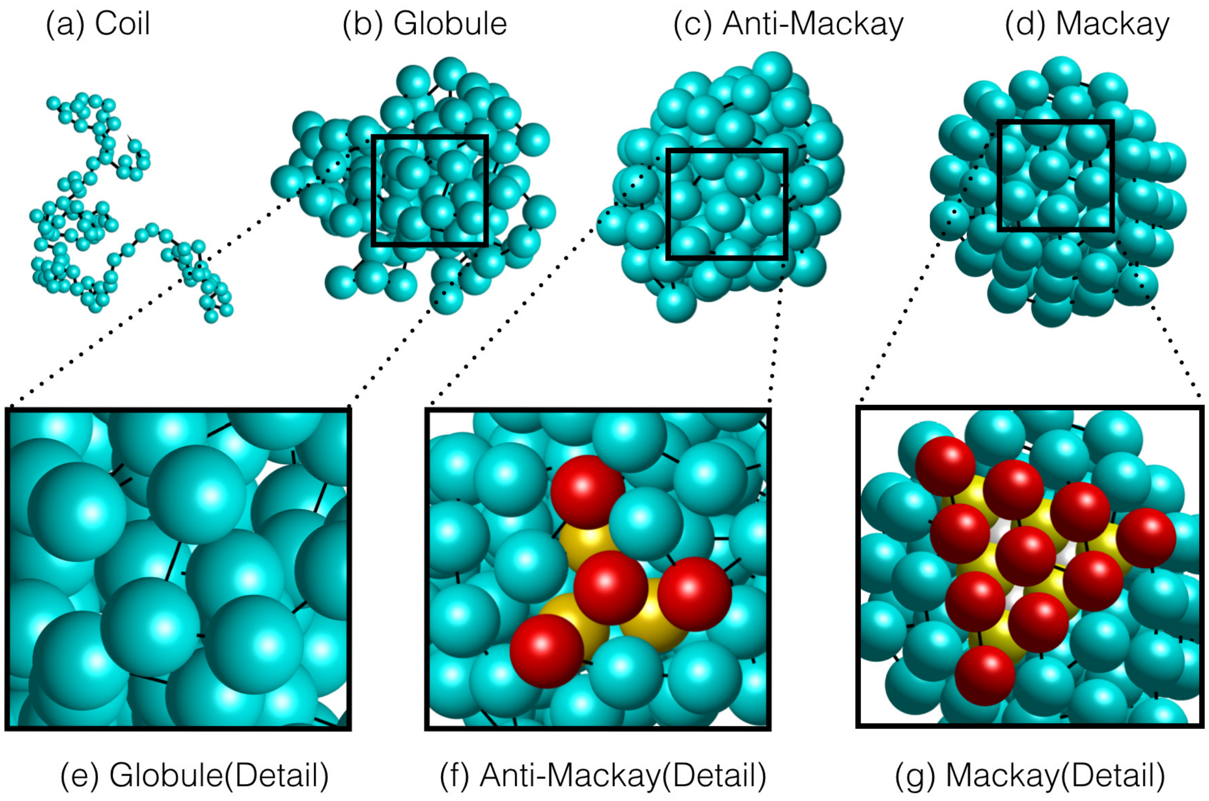

Figure 1.3: Three dimensional graphs of typical configurations of a homopolymer chain in (a) coil, (b) globular, (c) anti-Mackay, and (d) Mackay states, sampled by Monte Carlo methods, as the temperature or energy per monomer decreased. The $N = 102$ monomers are represented by blue spheres, except for those in in (f) and (g). Detailed stacking modes are shown in (e), (f) and (g): at the liquid-like globular state, the monomer positions are disordered, shown in (e); both anti-Mackay and Mackay states are crystalline, differ from each other by the monomer stacking symmetries, demonstrated by the red and yellow spheres in (f) and (g).

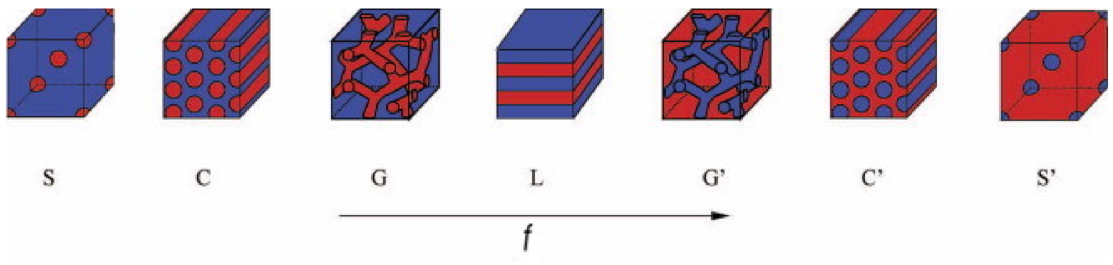

Figure 1.4: Microphase structures of diblock copolymers. These structures are labelled as, S which means body centred cubic spheres, C which means hexagonal cylinders, G which means gyroid, and L which means lamellar. Red is A rich region, in which the density of type A monomers is greater than type B monomers. Blue is B rich region. Taken from Ref. [103]

Another important example is the self-assembly of diblock copolymer melt. The equilibrium state of the diblock copolymer melt is determined by two parameters, f and χN. Here, χ is the Flory-Huggins parameter, which reflects the strength of the repulsive interaction, or incompatibility, between type A and B monomers. N is the total number of monomers along the diblock copolymer chain, i.e. polymerization. Similarly, at the high temperature regime, the domination of the entropy leads to a disordered, homogeneous phase, which means type A and B monomers distribute uniformly in the melt. When the melt is cooled to a temperature that is lower than a transition point, type A and B monomers will segregate from each other, and aggregate with the same type of monomers. Based on f, the diblock copolymer will self-assemble into different states. When f is small, type A monomers can only occupy a small portion of the space. They are clustered as microscopic spheres together. This structure is referred as the body centred cubic spheres. For the block copolymer with larger f, type A monomers can occupy more space. Different structures will be formed, such as hexagonal cylinders, where one type of monomers forms microscopic cylinders together; gyroid, where monomers form a bi-continuous structure. When f is near 0.5, type A and B form lamellar structures, which is a layered structure, as shown in Fig. 1.4.

1.3 Identifying polymer states

Polymer states could be studied very similarly as phases in condensed matter physics [6]. In condensed physics, a phase is a state of matter where all the chemical composition, thermodynamic properties, such as temperature, pressure, are uniform. An everyday example is solid, liquid and gas phase of water. When some thermodynamic quantities changed, one phase can transform into another phase, and this transformation is called phase transition [190], e.g. when temperature increases from below to above melting point, ice will melt into water.

The purpose of this section is to introduce how classical methods identifying states, to illustrate some inspirations physicists received from machine learning methods and to introduced basic ideas of machine learning methods.

1.3.1 Ehrenfest classification

The classification of phase transition was firstly developed by Paul Ehrenfest [43]. This scheme looks into the Gibbs free energy, which is written as a function of thermodynamic variables. A state corresponding to a local minimum of free energy is a meta-stable state, while a state that corresponds to the global minimum is the stable or equilibrium states. When examining the transition between states, this scheme examines the derivative of the free energy with respect to some thermodynamic quantities, such as temperature or pressure. The order of transition equals the lowest order at which the derivative of the free energy is discontinuous, and the discontinuous point is considered as the transition point. For example, Fig. 1.5 shows the free energy of liquid (red) and solid (blue) phase with respect to temperature. When the temperature is higher than T_C, the solid will have a lower free energy than the liquid phase, so solid will be the equilibrium state. Similarly, when the temperature is lower than T_C, the liquid will be the equilibrium state. In this example, the first order derivative of free energy is discontinuous at T_C. Therefore, the order of the transition is one, and the transition point is T_C.

1.3.2 Landau classification

The modern scheme for the classification of phase transitions was proposed by Lev Landau [92]. He introduced the concept of order parameter, which is defined as a thermodynamic function that has different values in different phases, and thereby could be employed to distinguish phases. Under Landau's scheme, the first order transition contains latent heat, which means the system absorbs or releases heat while the temperature stays the same. This means the heat capacity has a peak when the system is finite in size. During the transition, the system is in a "coexistence" state, in which some regions stay in one phase, and rest transformed into another phase, the order parameter will be discontinuous at the transition point. Second order phase transitions are also called continuous phase transition,

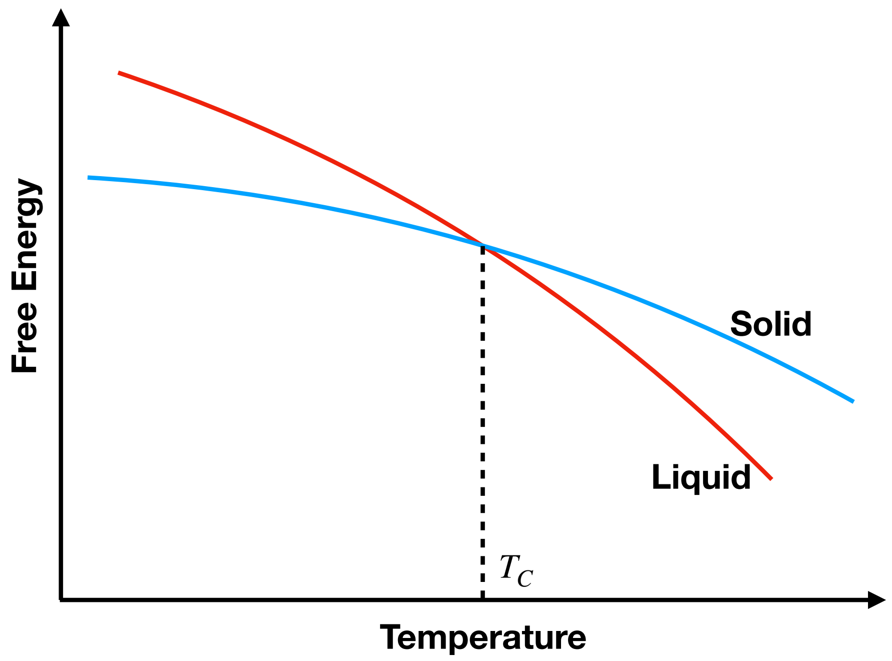

Figure 1.5: A sketch of the free energy of liquid (red) and solid (blue) phase with respect to temperature. T_C represents the transition temperature between the liquid and solid phase.

because the order parameter evolves continuously at the transition point. For a second order, the system is usually undergoing a change from a "random" state to a "ordered" state. This change will not require energy, therefore the transition happens smoothly. This means the heat capacity has an anomaly at the transition point. When the system is finite in size, we can see a peak as well.

Under this scheme, the identification of states of a homopolymer is performed with carefully designed order parameters. These order parameters could be evaluated analytically, measured experimentally or calculated numerically. In the numerical simulation, order parameters are calculated through a statistical average of samples generated from simulation methods, such as Monte Carlo methods [12] or molecular dynamics [56], etc. Because the coil and globule states are distinct in size, they can be characterized by the mean square radius of gyration, which is defined by:

$$\langle R_{\rm g}^2 \rangle = \left\langle \frac{1}{N} \sum_{i=1}^{N} (\mathbf{x}_i - \mathbf{x}_{\rm cm})^2 \right\rangle \tag{1.3}$$

where $\mathbf{x}_{\rm cm}$ is the centre of mass of a polymer, $\langle \cdots \rangle$ represents mean over multiple samples. The transition temperature, also referred as transition point, can be obtained by examining the specific heat, which is defined as:

$$C = \left(\left\langle E_{\rm int}^2 \right\rangle - \langle E_{\rm int} \rangle^2 \right) / k_B T^2 \tag{1.4}$$

where, k_B is the Boltzmann constant. The specific heat peaks at the transition temperature [81].

Traditionally, anti-Mackay and Mackay are two collapsed states that differ from each other only by how monomers are stacked together. They have a very similar size, which means it is impractical to use radius of gyration as the quantity to characterize the difference between these states. A new way to characterize their structural difference is required. Typically, anti-Mackay and Mackay state are characterized through bond orientational order parameters [193]. This parameter is firstly introduced by Steinhardt et al. to measure orientational symmetries in clusters [160]. To evaluate bond orientational order parameters, we start from associating a set of bonds connecting a monomer to its near neighbours. The spherical harmonics of some suitable set of bonds are averaged by

$$Q_l = \left(\frac{4\pi}{2l+1} \sum_{m=-l}^{l} \left| \frac{1}{N_b} \sum_{i=1}^{N_b} Y_{lm}(\theta_i, \phi_i) \right|^2 \right)^{1/2}, \tag{1.5}$$

where N_b is the number of suitable set of bonds, where "suitable" means the length of the bond is smaller than a threshold r_c [160]. Y_{lm} is a spherical harmonics of degree l, order m, and θ_i and ϕ_i are the polar angle and azimuthal angle of ith bond, respectively. These spherical harmonic functions expand bond orientations into mutually orthogonal directions. Different structures are expected to have different projections along these directions. For anti-Mackay and Mackay, we need to consider two types of averages. The first one is using all near neighbours in one cluster, i.e. every possible bonds in the cluster should be taken into account to obtain a global parameter, Q_6. The other one is just considering the bonds connecting the central particle with the particles that belong to the core of the cluster, corresponding to the local order parameter $Q_6^{\rm core}$ [193].

1.3.3 Machine learning methods

Using order parameters to identify states works well when we only need to identify a small number of states. Nevertheless, with the variation of polymerization, a homopolymer can exhibit plenty of different states [193, 148], much more than coil, globule, anti-Mackay and Mackay states we discussed. In order to distinguish all these states, one may have to design new order parameters, due to the limited generalization capability of the order parameters. To make things even worse, some of these order parameters could be very difficult to design. As more and more new states are discovered and need to be identified, using order parameters will be intractable. Therefore, it is desirable to have a method that can identify multiple states simultaneously without the necessity of designing order parameters. For this purpose, machine learning methods are considered to be good candidates. This idea naturally arises from one of the extensively studied machine learning examples: image recognition. The purpose of image recognition is to automatically identify or classify a large volume of pictures that belong to one of the multiple predefined classes.

In a typical application, a two dimensional picture is digitized into pixels, and the pixel's intensities, usually represent as a value within 0 and 1, are used as the inputs for training a machine learning model. In the training stage, one inputs pre-labelled pictures to the machine learning model. The label specifies which class does an input picture belongs to. For each input, The machine learning models will give an output that represents a prediction made by the machine learning model. The output of machine learning models could be very different from the label at the beginning of training. During the training, the outputs will gradually align with the label. After the machine learning models are well trained, the output will be fairly close to the label. For example, in a case when the pictures are pre-categorized as 4 classes, the label is usually written as a binary vector, i.e. a label $[0, 1, 0, 0]$ means this picture belongs to the second class. When we see an output $[0.01, 0.96, 0.02, 0.01]$, it means the machine learning model infers the probability of the input picture belongs to each classes as 0.01, 0.96, 0.02, 0.01 respectively.

Recently, physicists start to use machine learning to classify phases and to identify phase transition points [117]. Melko and coworkers bring this idea to study the phase transition of spin models [19]. We take the Ising model as an example. It has two phases: a disordered phase where each spin can point up or down randomly, and an ordered phase where each spin tends to align its direction with all neighbour spins. The spins have binary values like the pixel's intensity, and each configuration, regardless of dimension, can be considered as a "snapshot" of pixels as well. The ordered and disordered states of the Ising model can be labelled as $[0, 1]$ and $[1, 0]$ respectively. Then, one can train machine learning models based on the labelled configurations to classify the ordered or disordered phases [19]. Other similar findings are also reported around the same time, including producing the phase diagram of topological quantum phase transitions [195], determination of the critical temperature without introducing order parameters [170], etc.

From a machine learning point of view, polymeric systems have a certain level of similarity with the Ising model. These systems also have multiple states that need to be identified. These states of a polymer could be determined by the coordinates of the monomers, that can be utilized as the input of the machine learning methods. These similarities provide a compelling reason to use machine learning approaches to identify polymer states and transitions. The machine learning community has developed various methods for the purpose of classifying different states. In our study, we studied two methods, feed-forward neural network (FNN) and principal component analysis (PCA). These methods have different philosophies. The data set we have are usually stored in a data matrix. Each row of this matrix is a sample. The number of rows of this matrix is the number of training or test samples we have. The number of columns is the dimensionality of the training or test data. FNN generates the inference, i.e. the output of FNN, directly. The inference is determined by the FNN parameters. The training of an FNN adjusts its parameters so that the inference made by the FNN aligns with the label provided. Therefore, FNN will have the capability of classifying states after well trained. On the other hand, PCA tends to reduce the dimensionality of the data to 2 or 3 in a way that similar samples are mapped to the same place in the low dimensional representation. By reduction, we mean to find a function that maps the data matrix to a new matrix with fewer columns. This new matrix stores the low dimensional representation of the training data. Then, clustering analyses will be performed to the training data in the low dimension representation, to determine the class to which each training sample belongs. There are various ways to perform the dimensionality reduction. However, this thesis mostly focuses on PCA. Our efforts and results are discussed in chapter 3 and chapter 4.

It is worthwhile to mention that machine learning methods have been applied to many other studies in soft matter physics, such as evaluating the partition function of Lennard-Jones fluids [38], generating polymer conformations [192], identifying structural defects in polycrystalline [152] or liquid crystals [171], predicting the folding of a protein based on amino acid sequence or inverse the amino acid sequence from a folded structure [102], and so on.

1.4 Predicting equilibrium states of block copolymers

In the previous section, we demonstrated that the combination of machine learning methods and sampling methods leads to a great many successful studies. In typical examples, sampling methods generate configurational samples, and machine learning methods are trained to identify which state does these configuration belongs to. In this section, we are trying to deal with a different problem, that is we want to develop a machine learning method that can directly predict the equilibrium states of a given polymeric system.

1.4.1 Micro-phase separation

In phase transition, one phase turns into another phase, such as ice turns into water. In phase separation, a homogeneous mixture separates in two distinct phases. An everyday example is oil and water phase separation. The immiscibility between oil and water give them a tendency to occupy different regions in a container. When the temperature of oil and water mixture is high enough, entropy overcomes the immiscibility, oil and water will exist in a form of uniform emulsion. When the temperature is smaller than a transition point, the immiscibility drives oil and water to segregate into different regions, and create two phases from the homogeneous mixture. This phase separation happens macroscopically, for example, we can see it with naked eyes after a hot chicken broth was left inside a refrigerator for a while.

Micro-phase separation is very similar to oil and water phase separation. A common example is diblock copolymer melts. For this system, the driving factor of the phase separation is the incompatibility between type A and type B monomers. Because type A and type B blocks are covalently bonded, these monomers can not segregate into different regions macroscopically but form mesoscopic periodic structures. The polymer thermodynamics describe the incompatibility by the Flory-Huggins parameter and use χN as an indication of micro-phase separation happens or not. We have seen the different states diblock copolymer self-assembles into in Fig. 1.4, upon the combination of χN and f.

1.4.2 Traditional methods

There are multiple theories developed to predict the equilibrium state of a polymeric system, within which field-based simulations are widely adopted in the study of mesoscopic phases in melts of blends. The behaviour of polymers at the different length scale concerned can be easily varied according to the representation of field function [116]. In particular, the mesoscopic simulation to the ordered structures self-assembled from block copolymers in a characteristic feature size $10-100$ nm will highly benefit from this methodological flexibility, due to the desirable computational capacity in a broad range of structural length scale, in comparison with particle-based simulation. Another fascinating advantage of the field-based simulation is the exhibition of the relatively close connection with experiments, by sharing the common mesoscopic variables such as Flory-Huggins parameters and statistical segment length. By means of the mean field approximation in which only one configuration is assumed to be much more important than all other configurations, in practice, one can straightforwardly access the self-consistent field theory (SCFT) through the statistical field theory derivation. In the recent thirty years, the power of SCFT has been demonstrated particularly in the study of phases and phase behaviour of block copolymer systems. For example, SCFT was able to produce an accurate phase diagram of diblock copolymers that has a good agreement with experimental results [113], as shown in Fig. 1.6.

The origin of SCFT can be traced back to the work done by Edwards in the 1960s [42]. Later, Helfand explicitly extended this theoretical framework to the block copolymer in 1975 [62], and thereafter important contributions were made by Hong and Noolandi [67]. In order to explore the ordered meso-structures assembled by diblock copolymers, in 1980 Leibler [100] developed the analytical theory through expanding the free energy functional around the homogeneous background state, only valid in the weakly-segregated regime. Another analytical method, in the strong-segregated regime assuming that the polymer chain is strongly stretched, was developed by Semenov in 1985 [82]. The development of numerical solutions to the PDEs resulting from the SCFT pushed the research on the structures formed by the block copolymer forward. Matsen and Schick in 1994 [111] utilized the Fourier spectrum method to numerically solve the self-consistent equations and successfully obtained the phase diagram of self-assembly of diblock copolymers. The approach was later applied to a variety of polymeric systems, such as polymer blends [112], multi-block copolymers [115, 57, 191, 40], polymer brushes [37] and polyelectrolytes [110]. Another branch of numerical approaches, well known as real space algorithm in which a numerical solution is directly explored in the spatial space without prior knowledge about the symmetry of order structures was developed by Fredrickson et al. in 1999 [40]. By introducing the Gaussian fluctuations to the SCFT free energy functional, Shi et al. investigated the ordered structures affected by the fluctuation effects [154]. In addition, SCFT was also applied to successfully predict the complex phase behaviours of liquid crystalline polymers in bulk and confinement systems [20]. Besides, there is a large number of studies that significantly contributed to the development of SCFT as well. Many valuable reviews and books [5, 7, 48, 49, 58, 113, 116, 121, 153] shed substantial light on the applications of SCFT.

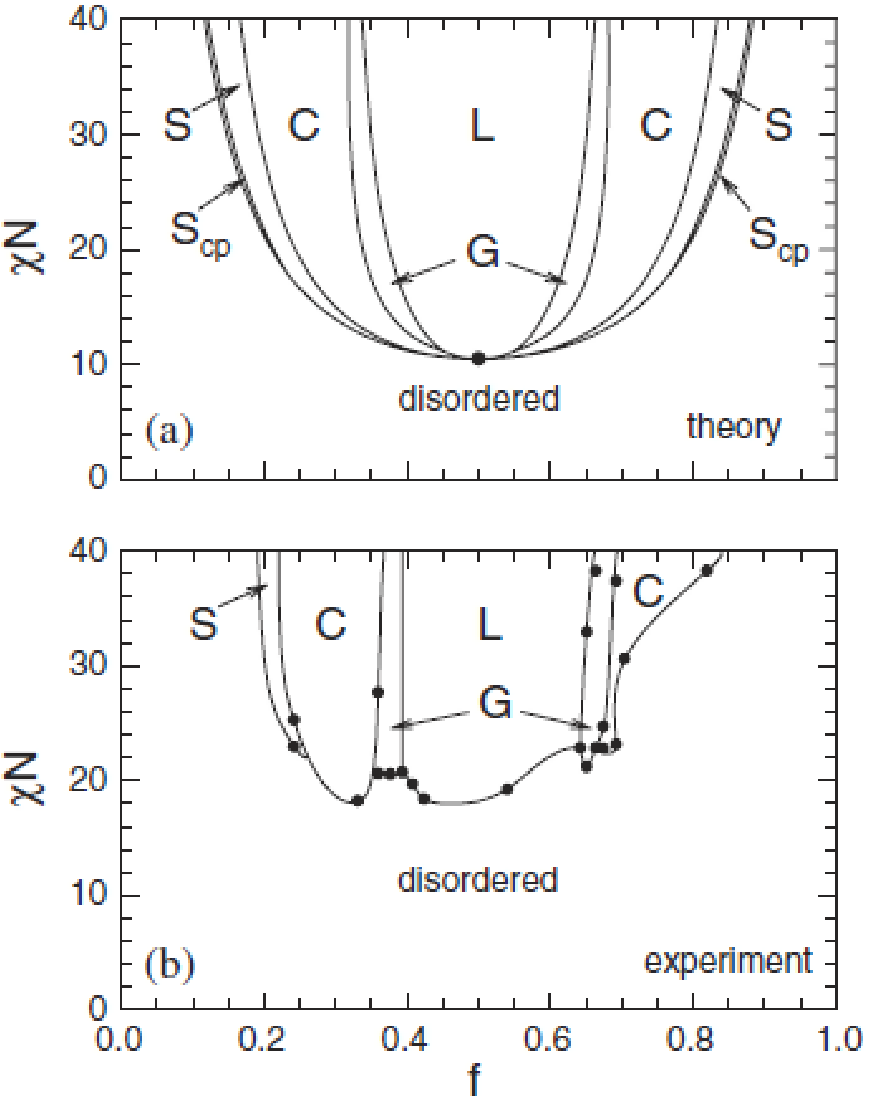

Figure 1.6: Phase diagram of diblock copolymer. (a) Theoretical prediction of the SCFT; (b) Experimental phase diagram measured using polystyrene-polyisoprene diblock copolymer. Taken from Ref. [113]

The SCFT scheme leads to a set of equations, called self-consistent equations, which are nonlinear coupled integrodifferential equations, that need to be solved self-consistently.

Traditionally, the self-consistent equations are solved through multiple loops starting from a guess of $W(\mathbf{r})$. The inner loop starts from solving the modified diffusion equations (MDEs) containing the current $W(\mathbf{r})$ iteratively based on numerical methods such as finite difference methods or spectral methods to obtain propagators of chains, where the propagators are the conditional probability of a monomer at spatial position $\mathbf{r}$, given the first monomer is at spatial position $\mathbf{r}'$. The intermediate loop updates the $W(\mathbf{r})$ based on the density profile calculated from propagators [48, 165, 114]. Here, the density is the statistical average of the microscopic monomer density. This loop stops when $W(\mathbf{r})$ converges after multiple iterations. Once the solution is obtained from these two loops, the free energy can be evaluated. On top of these loops, we need an outer loop to adjust the simulation box size, to find the optimal box size with minimum free energy. The solution of the self-consistent equations represents the equilibrium state when the free energy of this state is the global minimum or meta-stable states otherwise. The solution one obtains is highly dependent on the guess of $W(\mathbf{r})$ at the beginning. Therefore, the possible states of a polymeric system could be discovered through solving self-consistent equations with different initial guesses.

1.4.3 Machine learning based methods

The dimensionality of the self-consistent equations is determined by how many variables the propagator have. It changes based on the polymer models. One of the fundamental assumptions in SCFT is that the chain configuration can be represented by the Gaussian chain model (GSM). For this model, the dimensionality of the propagator, $q(x, y, z; t)$, that needs to be solved is four, where t is a time-like variable and another three are spatial variables. When the rigidity of a polymer chain is in concern, a more appropriate model is worm-like chain (WLC). In order to find the equilibrium states of worm-like diblock copolymers, the propagator, $q(x, y, z, u_x, u_y; t)$, that needs to be solved is a six dimensional diffusion like equation [77]. Similarly, t is a time-like variable; x, y, z are variables that specify a spatial point; u_x, u_y are coordinates of a unit vector that specify the direction of the monomer located at (x, y, z) pointed in. The efficiency of the SCFT is highly dependent by the solver of the MDEs. Unfortunately, the computational complexity of traditional methods, e.g. finite difference methods or spectral methods, grows exponentially with respect to dimensionality, which means it takes longer and longer for the solver to find an equilibrium state when the polymeric systems have an increasing number of parameters. This phenomenon is referred as the “curse of dimensionality”.

Machine learning provides a new perspective in searching for equilibrium states of a polymeric system. The foundation of this perspective is a machine learning based PDE solver. The idea is to exploit a basic property of an FNN, known as the universal approximation theorem, which states that any continuous functions can be effectively represented by FNNs, provided that adequate neuron nodes are used [69, 32, 68]. The FNN simply takes the variables of the functions as the inputs and it outputs functions themselves. The variety of functions are represented by the FNN parameters such as the weights and biases of the sigmoid functions that connect the neuron nodes [61]. If we can tune these FNN parameters to represent functions that satisfy differential equations and their auxiliary conditions, then a solution is considered found by us [89, 90]. This tuning is achieved by minimizing a cost function which embeds the targeted differential equations and their auxiliary conditions as squared modulus. At the moment when this method was proposed, i.e. around 2000s, due to the limited computational capability, most of the works focused on solving low dimensional PDEs by FNNs with a small number of hidden neurons [89, 90, 169, 134, 88]. Recently, taking advantage of the rapid growth of computational power, researchers start to focus on solving high dimensional differential equations with neural networks. For example, Karniadakis et al. [135] explored the accuracy and time consumption on solving nonlinear PDEs by neural networks with respect to a different number of hidden layers. Han et al. and E et al. [59, 41] proposed a deep learning algorithm that can efficiently solve high dimensional PDEs for time-dependent physical processes that involve stochastic input. One striking advantage of implementing machine learning technique to solve PDEs is the possibility of breaking the curse of dimensionality for a multi-variable problem [59, 156, 181, 76]. These works reflect the necessity of exploring the usage of neural networks on solving PDEs that used to describe high dimensional physical systems. Motivated by this necessity, we have designed an unsupervised, universal, machine learning based solver, with a particular focus on solving self-consistent equations and searching for polymer states. Our studies and results are discussed in chapter 5.

1.5 Objective of thesis

The objective of this thesis is to illustrate the machine learning methods we have applied to study polymer physics. To fulfill this purpose, we will describe our studies mainly based on two categories:

- identifying polymer states by supervised or unsupervised learning methods;
- utilizing FNNs to formulated a machine learning based PDE solver, that can be use to search for equilibrium states of diblock copolymer melt under SCFT scheme.

1.6 Organization of thesis

This thesis has 6 chapters: chapter 1 - research motivation and historical efforts; chapter 2 - introduce to the methodologies; chapter 3 - identifying polymer states by supervised learning; chapter 4 - identifying polymer states by unsupervised learning; chapter 5 - machine learning solver for the modified diffusion equation of AB diblock copolymer; chapter 6 - summary and outlook.

In chapter 2, we firstly introduce the machine learning methods that we utilized for identifying polymer states. Then, we show how to build a machine learning PDE solver for predicting block-copolymer structures.

In chapter 3, the ability of an FNN to learn and classify different states of polymer configurations is systematically explored. Performing numerical experiments, we find that a simple network model can, after adequate training, recognize multiple structures, including gas-like coil, liquid-like globular, and crystalline anti-Mackay and Mackay structures. The network can be trained to identify the transition points between various states, which compare well with those identified by independent specific-heat calculations. Our study demonstrates that the neural network provides an unconventional tool to study the phase transitions in polymeric systems.

In chapter 4, motivated by the high interpretability of unsupervised learning methods, we show the ability of principal component analysis (PCA) to distinguish polymer configurations and discover polymer phase transitions. PCA is applied to coil, globule, anti-Mackay and Mackay states to reduce the dimensionality of polymer configurations. The analyses of the polymer configurations in terms of low-dimensional representation not only identify the distinct states in the feature space, but also offer significant insights to understand the relation between salient features and order parameters in physics. In addition, a hybrid scheme combining the PCA and neural network is utilized to accurately locate the transition point between polymer states.

In chapter 5, we firstly present a universal, machine-learning based solver for partial differential equations in general. The solver approximates the target functions by FNNs and adjusts the network parameters to produce the approximation to the desirable solutions. The differential equations themselves, together with boundary conditions, etc, are treated in the cost function. We demonstrate the usage of the algorithm by solving diffusion equations in one dimension and high dimensions and found that the machine-learning solver breaks the curse of dimensionality expected in a high-dimensional case. We also demonstrate that the algorithm can be used to solve a set of much more complicated, coupled integrodifferential equations, encountered in predicting polymer microphase-separated structures, in terms of the self-consistent field theory.

In chapter 6, we concluded the discoveries of our researches, and proposed some possible future studies.

Chapter 2

Machine learning methods

2.1 Background

Machine learning is an active research branch of computer science, which has vastly matured in recent years [47, 179, 187, 120, 122, 79]. The term "machine learning" was created by A. L. Samuel [142] in 1959. Over the years, it was developed into a research field that using machine learning methods to construct a mathematical model from observed data, also called "training data", in order to perform a task on unseen data, also called "test data", without of given *a priori* knowledge of the system itself [120]. Nowadays, multiple machine learning methods, such as support vector machine [14, 163], random forest classifier [65, 66], neural network [123], have been developed. The tasks they can perform usually are classification or regression, and the capability of performing these tasks could be obtained from training the machine learning methods.

In a broad sense, classification is the task of categorizing training data and test data into classes. The training or test data is usually stored in a data matrix, shown as $\mathbf{X}$ in Fig. 2.1, in a way that each row, $\mathbf{r}_i = (x_{i1}, x_{i2}, \cdots, x_{iD})$, D-dimensional vector, is a sample. For example, if the sample is a configuration of a polymer with polymerization N, the x_{ij}'s are the coordinates of all the monomers, and $D = 3N$. The number of rows, M, of this matrix is the number of training or test samples we have. The number of the columns, D, is the dimensionality training or test data. The label of the data specifies which pre-defined classes do input samples belong to. A typical choice is to store them as a matrix as well. In Fig. 2.1, ν is the label matrix. The ith row of matrix ν is the label of the ith sample. The number of column K is the number of pre-defined classes. This task could be accomplished by supervised learning when the training data is labelled, or unsupervised learning when

$$\begin{tikzcd}
& \mathbf{X} = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_M \end{bmatrix}_{M\times D} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1D} \\ x_{21} & x_{22} & \cdots & x_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{M1} & x_{M2} & \cdots & x_{MD} \end{bmatrix}_{M\times D} \arrow[dl, "\text{Supervised}"'] \arrow[dr, "\text{Unsupervised}"] & \\
\nu = \begin{bmatrix} \nu_{11} & \nu_{12} & \cdots & \nu_{1K} \\ \nu_{21} & \nu_{22} & \cdots & \nu_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ \nu_{M1} & \nu_{M2} & \cdots & \nu_{MK} \end{bmatrix}_{M\times K} & & \mathbf{X}' = \begin{bmatrix} r'_1 \\ r'_2 \\ \vdots \\ r'_M \end{bmatrix}_{M\times D'} = \begin{bmatrix} x'_{11} & x'_{12} & \cdots & x'_{1D'} \\ x'_{21} & x'_{22} & \cdots & x'_{2D'} \\ \vdots & \vdots & \ddots & \vdots \\ x'_{M1} & x'_{M2} & \cdots & x'_{MD'} \end{bmatrix}_{M\times D'}
\end{tikzcd}$$

Figure 2.1: A graphical illustration of the data matrix, labels, supervised learning and unsupervised learning. The arrow with text “Supervised” means supervised learning methods tend to find a function that maps data matrix to labels. The arrow with text “Unsupervised” means unsupervised learning methods tend to find a way to transform the data matrix to a new data matrix with smaller dimensionality.

the training data is unlabelled. In supervised learning methods, we are keen to find a function that maps training data to its label. This function can be generated by many machine learning methods, but we mainly focus on the ability of classification of a feed-forward neural network (FNN). On the other hand, unsupervised learning methods tend to reduce the dimensionality of the training data to 2 or 3 in a way that similar samples are mapped to the same place in the low dimensional representation. By reduction, we mean to find a function that maps the data matrix, $\mathbf{X}$, to a new data matrix, $\mathbf{X}'$ with fewer columns, i.e. $D' < D$. This new matrix stores the low dimensional representation of the training data. Then, clustering analyses will be performed to the training data in the low dimension representation, to determine the class to which each training sample belongs. Similarly, there are various ways to perform dimensionality reduction. However, this thesis mostly focuses on exploring the capability of PCA.

In contrast, regression is the task of estimating the function relationships between training data, $\mathbf{r}$ and values, $\mathbf{y}$. The training data are commonly called dependent variables or regressors, and values are commonly called independent variables or predictors. For a regression model, the dependent variables and independent variables are assumed to have a function relationship as

$$\mathbf{y} = \mathbf{q}(\mathbf{r}, \theta) \tag{2.1}$$

In Eq, (2.1), θ are parameters that specifically determines a unique $\mathbf{q}$, to be obtained through training.

Regression can be performed both by "Classical" methods, such as linear regression or spectral decomposition [167] and by "Machine Learning" methods, such as FNN. The boundary between regression by "Classical" methods and by "Machine Learning" is blurry. As a rule of thumb, regression by classical methods usually restrict $\mathbf{q}$ to have some specific forms and θ can be calculated deterministically. For example, in ordinary linear regression, $\mathbf{q}$ are assumed to be linear functions, and θ are the slopes and intercept. It should be clear that θ can be calculated deterministically. When the spectral decomposition is used, $\mathbf{y}$ is expanded on a set of basis functions. The θ are the superposition coefficients of the expansion. In comparison, $\mathbf{q}$ in machine learning based methods usually are generic. As an example, when an FNN is used, θ are the weights and biases of the neural network. Given a different set of θ, $\mathbf{q}$ can take a variety of forms. These parameters need to be obtained iteratively through training.

2.2 Classification by supervised learning

2.2.1 Architecture of an FNN for classification

This subsection aims at illustrating the architecture of the FNN we used for the classification of polymer states. An FNN is a computational implementation of machine learning that has demonstrated surprising capability in recognizing patterns of enormous complexity, after appropriately trained by human or self-trained through learning mechanisms [13, 33, 53, 74, 127, 144, 150]. Originally motivated by the desire to establish an algorithmic model of the neuronal configuration of a mammalian brain, one finds that an artificial neural network with a very simple underlying structure can already successfully perform many complex tasks. For example, simple neural network models have shown transformative success in hand-writing and speech recognition [63, 97, 106, 123].

An FNN consists of three kinds of layers, input (i), hidden (h), and output (o). The neurons or "nodes" in input, hidden, and output layers are called input neurons, hidden

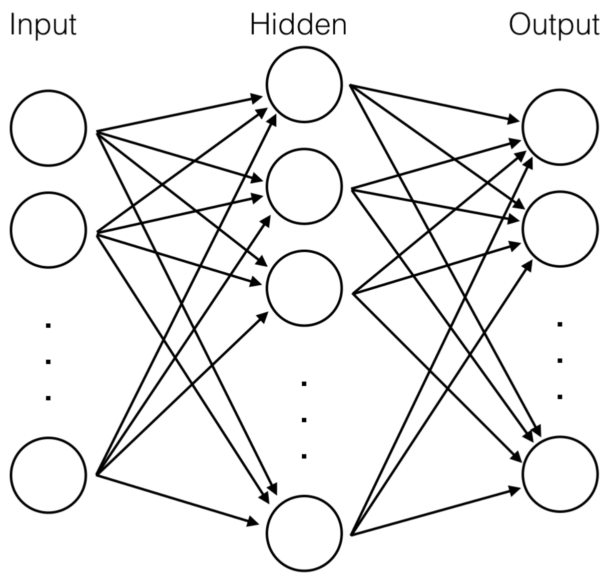

Figure 2.2: A graphical illustration of a fully connected FNN. The circles represent input, hidden, output neuron nodes, and arrows represent weights and bias. The neuron nodes are connected via various weights, bias, and activation functions layer by layer.

neurons and output neurons respectively. In this thesis, by default, the neural network we used contains one input layer, one hidden layer, and one output layer shown as Fig. 2.2. The input layer will have N_i neurons, and N_i needs to be set as the same as the dimensionality of the training data, i.e. $N_\mathrm{i} = D$. The hidden layer will have N_h neurons. The output layer will have N_o neurons, and N_o needs to be set as the number of classes to be classified or the number of functions to be regressed based on the anticipated tasks. These neurons are connected by arrows, representing the weights, $\mathbf{w}$, and biases, $\mathbf{b}$, of neural network, that forms a fully connected graph between, but not among the neurons beyond, consecutive layers. The weights and biases are also referred as the FNN parameters. The data will be fed into the neural network through input neurons. Then, information contained in data will be processed through hidden neurons and passed onto output neuron. Finally, the output neuron will provide an “inference”.

2.2.2 Supervised learning

Supervised learning algorithms aim at learning a function that maps training data to their label. Using the classification of the coil state and the globule state as an example, Fig. 2.3

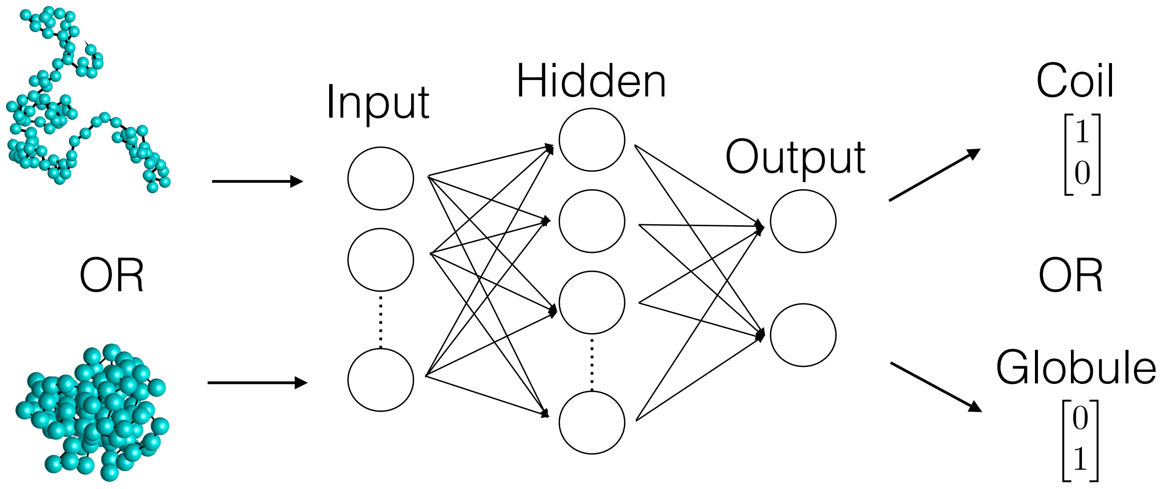
Input
Hidden
Output
Coil
$\begin{bmatrix}1\\0\end{bmatrix}$
OR
OR
Globule
$\begin{bmatrix}0\\1\end{bmatrix}$

Figure 2.3: A graphical illustration of supervised learning. In this example, there are two states, coil and globule, to be classified. The polymer configurations will be fed into FNN. The FNN will output an inference following Eq. 2.2 based on each input. The supervised learning algorithms aim at adjusting the FNN parameters so that the inference of FNN matchs the label provided.

illustrates how FNN classifies different polymer states via supervised learning algorithms. In supervised learning, the data consists of M polymer configurations to be classified, and their label. These configurations can be viewed as rows in a data matrix $\mathbf{X}_{M\times D}$. The rows of ν are labelled as $[1, 0]$ if they are coil states or labelled as $[0, 1]$ if they are globule states. Supervised learning methods start from a training stage. The training stage consists of consecutive epochs. At the beginning of an epoch, the M training samples, which is a $M \times D$ matrix, will be feed to FNN. For the ith training sample $\mathbf{r}_i = [x_{i1}, x_{i2}, \cdots, x_{iD}]$, i.e. a D by 1 vector, the output of the FNN is a vector function, $\mathbf{q}_i = [q_{i1}, q_{i2}, \ldots, q_{iK}]$:

$$\mathbf{q}_i = \sigma(\mathbf{w}_\mathrm{o} \cdot \sigma(\mathbf{w}_\mathrm{h} \cdot \mathbf{r}_i^T + \mathbf{b}_\mathrm{h})). \tag{2.2}$$

In Eq. (2.2) $\mathbf{w}_\mathrm{h}$ are the weights connect input and hidden layer, which is an N_h by D matrix. $\mathbf{w}_\mathrm{o}$ are the weights connect hidden an output layer, which is an K by N_h matrix; $\mathbf{b}_\mathrm{h}$ is the bias applied to the hidden neuron, which is an D by 1 vector. σ is the activation

Cost function	formula
Mean square error:	$J = 0.5\sum_{i=1}^{M}\sum_{j=1}^{K}(q_{ij}-\nu_{ij})^2$
Cross entropy:	$J = -\sum_{i=1}^{M}\sum_{j=1}^{K}\nu_{ij}\ln(q_{ij})$

Table 2.1: A list of different kinds of cost functions. $\mathbf{q}_{ij}$ is the jth FNN output of the ith sample, ν_{ij} is the corresponding label of the ith sample, j is the subscript of which this sample belongs to. K is the number of classed to be classifed.

function that has form

$$\sigma(z) = [1+\exp(-z)]^{-1}. \tag{2.3}$$

σ is an element-wise function, that is the shape of σ is determined by z. If z is a number, then $\sigma(z)$ is a number; if z is a vector or matrix, then σ is applied element wise on z. The output of FNN could be viewed as an inference about the input sample. If one sees an output $[0.7, 0.3]$, it means the FNN thinks the input is a coil with a probability 70%, and is a globule with a probability 30%.

The likelihood between the FNN inferences and labels could be measured by a cost function J. A smaller J means the current inferences given by FNN are closer to true labels of training data. J may take different forms such as mean square error or cross entropy loss, etc. as shown in Tab. 2.1. Typically, for classification, we use cross entropy cost

$$J = -\sum_{i=1}^{M}\sum_{j=1}^{K}\nu_{ij}\ln(q_{ij}). \tag{2.4}$$

The choice of cross entropy is due to practical reasons. During training, $\mathbf{q}$ will gradually approach the label. The cross entropy will not lead to a gradient vanishing problem as square error do.

Then, stochastic gradient descent [141] is performed to minimize J by adjusting weights and bias according to

$$\begin{aligned}\mathbf{w} &\leftarrow \mathbf{w} - \eta\nabla_{\mathbf{w}}J \\ \mathbf{b} &\leftarrow \mathbf{b} - \eta\nabla_{\mathbf{b}}J\end{aligned} \tag{2.5}$$

In Eqs. (2.5), $\nabla_{\mathbf{w}}J$ is the gradient of J with respect to all the weights. $\nabla_{\mathbf{b}}J$ is the gradient of J with respect to all the biases. η is a small quantity, called learning rate, that adjusts how much do we want $\nabla_{\mathbf{w}}J$ and $\nabla_{\mathbf{b}}J$ to change weights and biases. This is one epoch. After iterating through an appropriate number of epochs, the FNN is considered trained.

In desired cases, a trained FNN is expected to make correct classification on test data. Here, by "appropriate ", "desired", we mean the cases that training is not severely affected by overfitting [158].

Once the neural network is trained, we can feed in a batch of test samples. The output of each individual test sample is an inference made by the FNN. For example, if the inference is $[0.98, 0.02]$, then the input test sample should be labeled as a coil. We can compare the label inferred by FNN and the label associated with test data. The accuracy of the trained FNN is calculated as the number of correct inference over the total number of test samples.

Note, throughout this thesis, the FNNs introduced only have one hidden layer. I have also experimented with FNNs that have multiple hidden layers. Under the same requirement of accuracy, FNNs with multiple hidden layers take significantly less training time to reach the required accuracy. Using the same training time, FNNs with multiple hidden layers usually return higher accuracy results. These findings are as expected.

2.3 Unsupervised learning

Albeit supervised learning algorithms have shown great capability for classifying complex structures, they usually have limited interpretability. For example, it is hard to find which feature, or combination of features, does the FNN extracts from the training data to perform the classification during the training stage. However, these features are important to physicists as they can characterize the difference between structures, and they might be used as the order parameters. In addition, in many occasions, compared with unlabelled data, the labelled data are difficult or expensive to obtain, so we may not be able to collect enough training samples to train supervised learning algorithms. In comparison, unsupervised learning algorithms offer a way to analyze unlabelled data and in the meantime self-discover features that lead to the classification of data.

Generally speaking, the purpose of unsupervised learning algorithms include, but not limit to, dimensional reduction and clustering [13]. The goal of dimensional reduction is to find a map from the original representation $\mathbf{r} = (x_1, x_2, \ldots, x_D)$ to a low dimensional representation $\mathbf{r}' = (x_1, x_2, \ldots, x_{D'})$, where $D > D'$. Clustering aims at finding a series of clusters in the low dimensional space. When the samples are clustered, we expect the samples in the same group share more commonalities than those in different clusters. Therefore, each cluster can be viewed as one class "discovered" by unsupervised learning algorithms. Some classical unsupervised learning algorithms, such as principal component analysis (PCA) [52], autoencoder [105], K-means [73], etc. can be utilized to classify polymer samples. In this thesis, we mainly focus on the application of PCA in polymeric systems.

PCA [128, 78] is an approach widely used to extract the features from a data set by virtue of a linear dimensionality reduction technique. The so-called principal components refer to a few of mutually orthogonal normalized vectors. They are defined in a way that the projection of data along the first principal component has the largest variance, the projection of data along the second principal component has the second largest variance, and so on. PCA finds principal components by conducting the diagonalization to the empirical covariance matrix calculated from the data matrix in the original coordinates. Usually, PCA is used for pattern recognition and image compression. Therefore, PCA could be adopted to investigate the significant variation of polymer configurations changing with temperature and to play a role as an indicator of structural transition for polymers.

Suppose there are M polymer configurations to be classified. These configurations can be viewed as data points in D-dimensional space, where $D = 3N$. After subtracting the individual mean values to all data in each row, one obtains a data matrix $\mathbf{X}_{M\times D}$ that the average of all the elements in the same row is zero. Implementing a standard orthogonal transformation to the covariance matrix $\mathbf{X}^T\mathbf{X}$, i.e.

$$\mathbf{X}^T\mathbf{X}W_l = \lambda_l W_l, \tag{2.6}$$

PCA derives a series of eigenvectors $\{W_l\}_{l=1}^{D}$ sorted by its corresponding $\{\lambda_l\}_{l=1}^{D}$ in descending order, i.e. $\lambda_1 \geq \lambda_2 \geq \lambda_3 \cdots \geq \lambda_D$. Here, W_l is an M by 1 matrix. Due to the positive semi-define property of the covariance matrix, all λ_l are greater than or equal to zero. In practice, it is more convenient to work with normalized eigenvalues of the correlation matrix:

$$\tilde{\lambda}_i = \lambda_i \bigg/ \sum_{j=1}^{D} \lambda_j \tag{2.7}$$

One can obtain a column-wise orthogonal matrix $\mathbf{W}_{D\times D'} = (W_1, W_2, W_3, \cdots, W_{D'})$ by selecting D' ($D' \leq D$) principal components. Usually, the eigenvalues of the D'th principal component should be much larger than the other eigenvalues. These D' principal components are often referred as dominant principal components. Then, the optimal low-dimensional encoding of data $\mathbf{X}_{M\times D'}$ is given by

$$\mathbf{Z}_{M\times D'} = \mathbf{X}_{M\times D}\mathbf{W}_{D\times D'} \tag{2.8}$$

which is an orthogonal projection of the data onto the column space spanned by the eigenvectors. So, PCA is usually used to be a dimension reduction approach effectively extracting the major variation of data. This point of view can be justified by examining the correlation matrix of $\mathbf{Z}_{M\times D'}$, i.e.

$$\begin{aligned}\mathbf{Z}^T_{M\times D'}\mathbf{Z}_{M\times D'} =& \mathbf{W}^T_{D\times D'}\mathbf{X}^T_{M\times D}\mathbf{X}_{M\times D}\mathbf{W}_{D\times D'} \\ =& \mathbf{W}^T_{D\times D'}\mathbf{W}_{D\times D'}\mathbf{Diag}\mathbf{W}^T_{D\times D'}\mathbf{W}_{D\times D'}=\mathbf{Diag},\end{aligned} \tag{2.9}$$

where **Diag** is a diagonal matrix with diagonal elements $\lambda_1, \lambda_2, \cdots, \lambda_{D'}$. Looking backward at the mathematical operations mentioned above, one can straightforwardly interpret PCA as a technique of finding the main directions of maximizing the variance of the projected data.

2.4 Machine learning PDE solver

2.4.1 Regression by an FNN

The machine learning PDE solver is essentially a regression algorithm. Before we dive into the solver itself, in this subsection, we introduce how to use an FNN to perform regression tasks. Given a selected coordinate, $\mathbf{r} = (r_1, r_2, \cdots, r_D)$, one can assume that a vector function $\mathbf{q} = (q_1, q_2, \cdots, q_K)$ is generated by an FNN following the form:

$$\mathbf{q} = \mathbf{v}\cdot\sigma(\mathbf{w}\cdot\mathbf{r}+\mathbf{b}), \tag{2.10}$$

where $\mathbf{w}$, $\mathbf{b}$, $\mathbf{v}$ are FNN parameters to be determined, D is the dimensionality of the coordinate, and K is the dimensionality of the vector function. One advantage of the form in Eq. 2.10 is that it can represent almost any continuous functions [69, 32, 68]. The function relationship between $\mathbf{y}$ and $\mathbf{r}$ is assumed to follow Eq. 2.10. One needs to find an appropriate set of FNN parameters through training, so that $\mathbf{q}$ maps $\mathbf{r}$ to $\mathbf{y}$. The training stage of regression is very similar to the training stage of supervised learning. By constructing a cost function following

$$J = \frac{1}{2}\sum_{i=1}^{K}\left\langle (y_i - q_i)^2\right\rangle, \tag{2.11}$$

where the average bracket $\langle\cdots\rangle$ is performed on the M selected samples, we can measure the difference between FNN generated value, $\mathbf{q}$, and observed value, $\mathbf{y}$. Upon the minimization of J, one can find a suitable version of FNN parameters. Therefore, once J is minimized, Eq. (2.10) is considered as a function regressed based on the training samples. Given a set of unseen coordinates, one can predict the values by simply plugging these coordinates into Eq. (2.10).

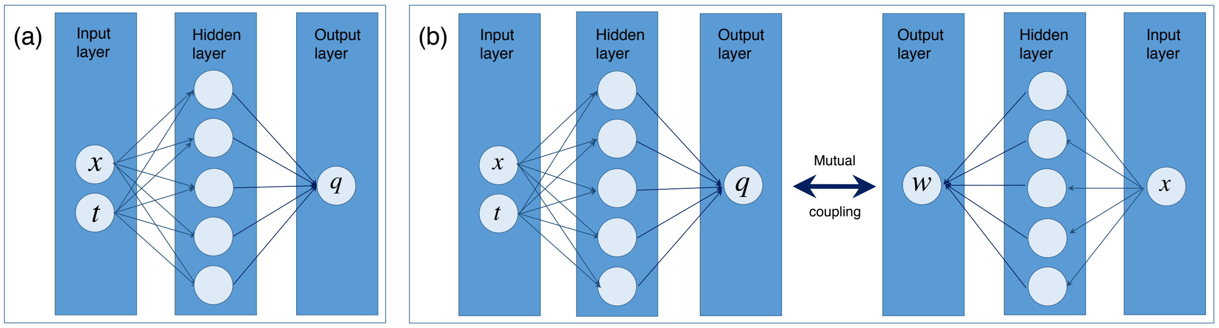

Figure 2.4: A graphical illustration of the FNN architecture of the machine learning PDE solver. The blue background is to emphasize that these FNNs are used for solving PDE instead of classification. All circles represent neuron nodes, where the input layer consists of nodes that have variables as input and the output layer are simply the functions to be determined. The connections between the input and hidden layers are assumed to be sigmoid functions and the connections between the hidden and output layers are assumed to be linear with adjustable coefficients. The architecture varies depending on the problem that needs to be solved. Two examples are shown here. (a) a simple two dimensional diffusion equation solver where $q(x;t)$ is the density of the diffusing material in an external field at location x and time t and (b) complicated, coupled modified diffusion equations where $q(x;t)$ is the density of the diffusing material that couples to an unknown external field $w(x)$. In both examples, the functions to be found are represented by FNN.

2.4.2 Constructing a machine learning PDE solver

In polymer physics, many phenomena are modelled as the solution of some partial differential equations (PDEs) with some given initial or boundary conditions [28, 48, 116]. One example we have seen is that the propagator of diblock copolymer melt can be solved from modified diffusion equations. The efficiency and accuracy of the PDE solver highly determine whether and how well we can study these phenomena. As the main purpose of this section is to explain the construction and training of a machine learning PDE solver, assuming that the solution we are interested, in this section, is a two dimensional function $q(x;t)$ on domain of interests $x \in [0, L], t \in [0, 1]$. The values of $q(x;t)$, that corresponding to **y**s a in regression task, are not directly observed. The Machine learning PDE solver aims at finding $q(x;t)$ with neural networks by taking the coordinates sampled from the domain of interests as the only input.

To begin with, we need to construct a solver based on the problem to be solved. Fig. 2.4 demonstrates two typical architectures we used in our studies. For the boundary value problem in this section, the machine learning PDE solver uses an FNN, as shown in Fig. 2.4(a), with two input nodes, i.e. $N_i = 2$, one hidden layer with N_h hidden nodes and one output neuron, i.e. $N_0 = 1$, to generate $q(x;t)$. In this case, $\mathbf{r}$ in Eq. (2.10) takes the form $\mathbf{r} = (x,t)$, and $q(x;t)$ takes the same form as in Eq. (2.10). With different choices of $\mathbf{w}$, $\mathbf{b}$, and $\mathbf{v}$, the machine learning PDE solver will generate different $q(x;t)$. Therefore, $q(x;t)$ could be viewed as a trial function. The universal approximation theorem guarantees that there exists a set of appropriate weighs and biases so that $q(x;t)$ generated by the FNN could be arbitrarily close to the solution of the boundary value problem. Now, the problem turns into how to find the appropriate weights and biases.

In order to find the optimal weights and biases, as values of $q(x;t)$ are not known to the machine learning PDE solver, a new cost function need to be designed. Assuming that $q(x;t)$ is determined by a boundary value problem represented as a series of operators:

$$\hat{D}(q(x;t)) = 0, \quad \hat{B}_1(q(x;t)) = 0, \quad \hat{B}_2(q(x;t)) = 0, \dots \quad \hat{C}(q(x;t)) = 0, \dots \tag{2.12}$$

Here, the number of operators varies based on the problem itself.

$\hat{D}$ is the differential operators acting on $q(x;t)$. The specific form of $\hat{D}$ depends on the PDE itself, for example, for a modified diffusion equation, it takes the form $\hat{D} = \partial/\partial t - \partial^2/\partial x^2 - w(x)$, where $w(x)$ is a given external field. When the analytic expression of $w(x)$ is given, we can use the architecture shown in Fig. 2.4(a) to solve for $q(x;t)$ directly. In the cases when $w(x)$ is unknown but can be obtained iteratively through some auxiliary conditions, we need to use another neural network to generate $w(x)$, and couples two neural networks together by $\hat{C}(q(x;t), w(x))$. The specific form of $\hat{C}(q(x;t), w(x))$ is determined by the auxiliary conditions. In this case, we need to use the architecture as shown in Fig. 2.4(b).

$\hat{B}$ will have specific forms depends on boundary conditions or initial conditions. For the first type of boundary condition, we have

$$\hat{B}_1(q(x;t)) = q(0;t) - b_1(t) = 0; \quad \hat{B}_2(q(x;t)) = q(L;t) - b_2(t) = 0, \tag{2.13}$$

where $b_1(t)$ and $b_2(t)$ are the value of $q(x;t)$ at boundaries. The second type of boundary condition can be implemented as

$$\hat{B}_1(q(x;t)) = \left.\frac{\partial q(x;t)}{\partial x}\right|_{x=0} - b_1(t) = 0; \quad \left.\frac{\partial q(x;t)}{\partial x}\right|_{x=L} - b_2(t) = 0, \tag{2.14}$$

in this case, $b_1(t)$ and $b_2(t)$ are the derivative of $q(x;t)$ at boundaries. The form of $\hat{B}$ for the third type boundary condition can be defined based on Eq. 2.13 and Eq. 2.14 very similarly. Periodic boundary condition requires $q(x+L;t) = q(x;t)$. If $q(x;t)$ is smooth, this condition also indicates $\partial^n q(x+L;t)/\partial x^n = \partial^n q(x;t)/\partial x^n$, where n is the order of partial derivative In practice, we find machine learning PDE solver can impose periodic boundary condition accurately by using two terms

$$\hat{B}_1(q(x;t)) = q(0;t) - q(L;t) = 0; \quad \hat{B}_2(q(x;t)) = \left.\frac{\partial q(x;t)}{\partial x}\right|_{x=0} - \left.\frac{\partial q(x;t)}{\partial x}\right|_{x=L} = 0, \tag{2.15}$$

where $\hat{B}_1$ imposes q to have equal values at boundaries, and $\hat{B}_2$ imposes the value of the first order derivative of q to be the same at the boundaries. In the cases $\hat{B}_3$ need to impose initial condition $q(x;0) = \psi(x)$, we have

$$\hat{B}_3(q(x;t)) = q(x;0) - \psi(x) = 0. \tag{2.16}$$

The difference between $q(x;t)$ and the solution of the boundary value problem could be measured by a cost function defined as

$$J = \frac{\alpha}{2}\left\langle \left|\hat{D}(q(x;t))\right|^2 \right\rangle + \sum_i \frac{\beta_i}{2}\left\langle \left|\hat{B}_i(q(x;t))\right|^2 \right\rangle + \frac{\gamma}{2}\left\langle \left|\hat{C}(q(x;t))\right|^2 \right\rangle \tag{2.17}$$

where α, β_i γ_i are penalty coefficients to emphasize the relative priority of each term in the cost function, that is under the same J, $q(x;t)$ tend to satisfy better with the terms with larger coefficients. When $J = 0$, all operators for this boundary value problem are satisfied, and $q(x;t)$ is considered as the solution of this boundary value problem. Hence, if we start from a guess of weights and biases, and minimize J, then the appropriate set of weights and biases is considered as found when J is smaller than a threshold.

2.4.3 Training the machine learning PDE solver

The steps on how to train a machine learning PDE solver are shown in Fig. 2.5(a)-(d). In step (a), the parameters of machine learning PDE solver are initialized randomly. Denoting the variable vector as $\mathbf{r} = (r_1, r_2)$[1], the (r_1, r_2) pairs are considered as training sample. Some of samples will be selected within the domain of interests for the evaluation of $\hat{D}$, $\hat{C}$ and some sample will be selected from the boundary for the evaluation of $\hat{B}$.

The activation function of the jth hidden node is a typical sigmoid, and for the i'th sample, it could be calculated as

$$\sigma_j = [1 + \exp(-z_j)]^{-1} \tag{2.18}$$

[1]In this notation, r_1 is the spatial coordinate x and r_2 is the time variable t.

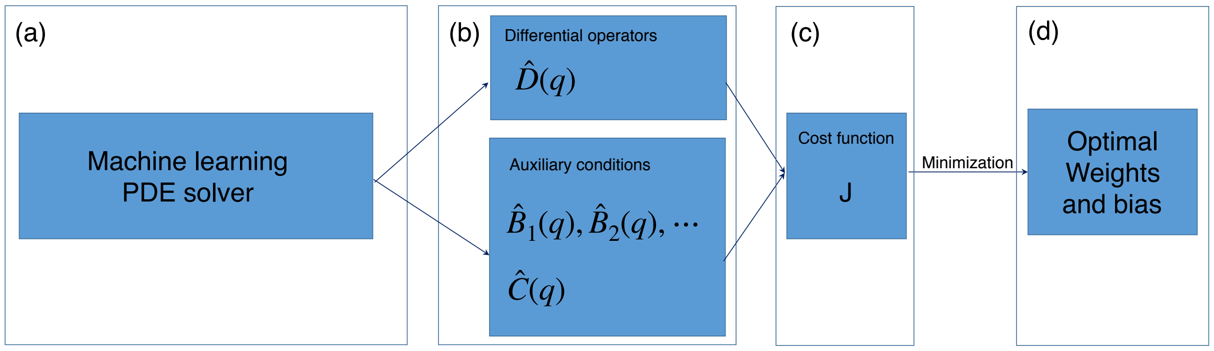

Figure 2.5: A flowchart of the training approach of the machine learning PDE solver. The training contains 4 steps. Step (a), we need to initialize the machine learning PDE solver with a guess of weights and biases. After a batch of training samples are fed into the solver, it will output the values of $q(x;t)$. Step (b), operators are calculated based on these values analytically. Step (c), cost function J is evaluated based on the sum of square module of operators. Step (d), the appropriate weights and biases can be obtained after J is minimized.

where $z_j = \sum_i w_{ji} r_i + b_j$ and all w_{ji} form an $N_h \times N_i$ parameter matrix and all b_j form an N_h-dimensional parameter vector. The output layer is connected to the hidden nodes by

$$q(\mathbf{r}) = \mathbf{v} \cdot \sigma \tag{2.19}$$

where $\mathbf{v}$ is an N_h-dimensional parameter vector and σ is formed by the elements defined in Eq. (2.18).

Once $q(\mathbf{r})$ is obtaind, we will move to step (b) to evaluate the operators. It is instructive to realize that all operators and derivatives needed can now be analytically determined. For example, the first order derivative in $\hat{D}$ can be calculated as

$$\frac{\partial q(\mathbf{r})}{\partial r_k} = \sum_j \frac{v_j \exp(-z_j) w_{jk}}{[1+\exp(-z_j)]^2}. \tag{2.20}$$

Second partial derivatives can be taken as well but the analytic expressions are omitted here. A through example calculation are provided in appendix. B. In short, the $\hat{D}$, $\hat{B}$, $\hat{C}$ can all be expressed analytically and calculated if the FNN parameters are known.

Step (c) consists of multiple epochs. Within an epoch, samples are selected from the variable space and used to evaluate the cost function J in Eq. 2.17. Then, in order to minimize J, the direction of the deepest descent in the FNN-parameter space needs to be calculated. Assuming that our machine learning PDE solver have only one $\hat{D}$, $\hat{C}$, and multiple $\hat{B}$, the gradient with respect to parameter could be written in a way as

$$\begin{aligned}\nabla_{\mathbf{w}} J =&\alpha \left\langle \left[\hat{D} q(\mathbf{r})\right] \left[\hat{D} \nabla_{\mathbf{w}} q(\mathbf{r})\right] \right\rangle \\ &+ \sum_i \beta_i \left\langle \left[\hat{B}_i q(\mathbf{r})\right] \left[\hat{B}_i \nabla_{\mathbf{w}} q(\mathbf{r})\right] \right\rangle \\ &+ \gamma \left\langle \left[\hat{C} q(\mathbf{r})\right] \left[\hat{C} \nabla_{\mathbf{w}} q(\mathbf{r})\right] \right\rangle\end{aligned} \tag{2.21}$$

The gradient of other FNN parameters, **b** and **v**, are calculated in a similar way. The back-propagation method is used to efficiently evaluate the required gradient on q [141]. According to the deepest descent method, the parameter vectors are then updated according to

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} J \tag{2.22}$$

$$\mathbf{b} \leftarrow \mathbf{b} - \eta \nabla_{\mathbf{b}} J \tag{2.23}$$

$$\mathbf{v} \leftarrow \mathbf{v} - \eta \nabla_{\mathbf{v}} J \tag{2.24}$$

where η is the learning rate. Here, the average bracket $\langle \cdots \rangle$ is performed on the selected samples.

Once J is minimized, an version of optimal **w**, **b** and **v** are considered found by the machine learning PDE solver.

Chapter 3

Identifying polymer states by supervised learning

3.1 Introduction

For physicists, one of the most familiar systems is molecular systems, in which various computer simulation algorithms are dedicated to study their properties. As reviewed in Section 1.3, traditional methods on identifying states are deterministic in nature, one need to design order parameters for all the states that need to be identified. These methods demand a deep understanding of the system and require advanced mathematical tools [183]. In contrast, machine learning methods are empirical and data-driven, which means they are able to learning the dependency between states, i.e. training samples, and label or able to extract the salient features without understanding the system. One of the most successful machine learning methods is FNN. Ref. [95] outlines the way to construct, train an FNN to fulfill the purpose of imagine recognition. It demonstrates that FNN has strong capability on classifying hand-written digits, everyday objects, such as balls, pets, cars, etc.

These applications have inspired physicists who work on condensed matter physics. Melko, et al. [19] set the foundation of studying state of matter by machine learning. They demonstrated an FNN trained by large volume of spin model configurations generated by Monte Carlo method can identify different states and transition points simultaneously. This approach investigates the average output values of the test samples generated from different temperatures. These values will form curves and their intersects are interpreted as transition points. One advantage of this approach is that only spin configurations are needed as input, without the need for system information, such as Hamiltonian, heat capacity, etc. Therefore, this idea is soon employed to identify the states sampled from other condensed matter models, such as Hubbard model [22], Haldane model [195], Heisenberg model [143], Potts model [101], etc. These applications justified the versatility and genericity of machine learning based methods.

This comes along with another wave of works. For example, Arsenault et al. used machine learning as a Green's function solver in dynamical mean-field theory for providing an efficient, low cost solver of the Anderson impurity model [2]. Behler et al. adopted a neural network to represent potential energy surface and force-fields to improve the accuracy of molecular dynamics [10]. Snyder et al. used a neural network to generate highly accurate density functional for electronic structure calculation [157]. Geiger et al. formulated a local structure detector in polymorphic systems [54].

Soft matter systems, especially coarse-grained models, have certain level of similarities with condensed matter systems. Therefore, motivated by the successes in condensed matter physics, in this chapter, we are motivated to explore the ability of an FNN on identifying various configurations sampled from Monte Carlo simulations of polymeric models. The configurations to be identified could be disordered, such as coil, partially-ordered, such as liquid-like globular, and ordered, such as two crystalline structures, called anti-Mackay and Mackay, in the low-energy regime. Indeed, these structures we listed here are well studied traditionally [148, 193]. Therefore, they are a good start for the exploration of the capability of hybrid traditional simulation methods, such as Monte Carlo methods, and machine learning techniques, such as supervised or unsupervised learning methods to study phase transitions in a classical polymeric system, and it is convenient to made comparisons between the results from traditional methods and machine learning methods.

3.2 Polymer models and states

In this chapter, we focus on a single homopolymer chain. In total, two polymer models, call GSM and FENE, are studied here. Both models have polymerization $N = 102$, and the monomers interacted with each other through bonded and non-bonded interaction. The reason we choose to study these two models is their states and transition points are well studied. It straightforward to compare the results from traditional methods and machine learning methods.

Within the first model, GSM, the bonded interaction between consecutive monomers is modelled as harmonic string potential, identical to the one used in the Gaussian chain model with a Kuhn length a [39]. Here, Kuhn length is a statistical segment length scale that represents the distance between consecutive monomers in a flexible polymer chain. The reduced bonded Hamiltonian of the polymer is

$$\beta E_{\text{bond}} = \frac{3}{2a^2} \sum_{i=1}^{N-1} \left(\mathbf{r}_i - \mathbf{r}_{i+1}\right)^2, \tag{3.1}$$

where $\beta = 1/k_{\text{B}}T$ is the Boltzmann factor, $\mathbf{r}_i$ is the coordinate of the monomer-i. The system interaction potential energy is

$$E_{\text{int}} = \frac{1}{2} \sum_{ij} U(|\mathbf{r}_{ij}|) \tag{3.2}$$

where $\mathbf{r}_{ij}$ is the distance vector between monomer-i and monomer-j. The non-bonded interaction, $U(\mathbf{r})$, between two monomers is a square-well potential: below a square distance of $0.81a^2$ the monomers repel through the excluded-volume interaction, and between $0.81a^2$ and $2a^2$ the monomers experience an attraction of magnitude $-\epsilon$, i.e.

$$U(r) = \begin{cases} \infty & \text{if } 0 \le r^2 < (0.9a)^2, \\ -\epsilon & \text{if } (0.9a)^2 \le r^2 \le 2a^2, \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

With the decreasing of temperature, GSM will exhibit a well-known coil-to-globule transition. Metropolis Monte Carlo simulations that incorporate the Boltzmann weight were used for this model to produce 5×10^3 independent configurations at every specified temperature $k_{\text{B}}T/\epsilon$ where k_{B} is the Boltzmann constant. The detailed information of the Monte Carlo procedure is documented in Appendix A.3.1. Assessing the data shown in Fig. 3.2(a) for both reduced mean-square radius of gyration, $S^2 \equiv \langle R_{\text{g}}^2 \rangle / a^2$, and mean-square deviation of the total energy from its average (which is proportional to the specific heat), $\tilde{C} \equiv [\langle E_{\text{int}}^2 \rangle - \langle E_{\text{int}} \rangle^2]/N^2\epsilon^2$, we observe that a coil-to-globule phase transition takes place at $k_B T/\epsilon \approx 2.0$, corresponding to the location of the peak in $\tilde{C}$.

Within the second model, FENE, the bonded monomers are connected by a particular implementation of the Finitely Extensible Nonlinear Elastic (FENE) interaction, and the non-bonded monomers interact with each other in the form of truncated Lennard-Jones (LJ) potential with a potential well-depth $-\epsilon$. Mathmetically, this model has a system energy as the sum of the bonded and interaction energies. The former is described by

$$E_{\text{bond}} = \sum_{i=1}^{N-1} E_{\text{FENE}} \left(|\mathbf{r}_i - \mathbf{r}_{i+1}|\right), \tag{3.4}$$

where E_{FENE} is a particular realization of the FENE model,

$$E_{\mathrm{FENE}}(r) = -20\epsilon R^2 \ln[1 - (r - r_0)/R)^2], \tag{3.5}$$

in which r is the distance between two connected monomers, $R = 0.3b$ controls the bond-length variations, and $r_0 = 0.7b$. The interaction potential energy formally follow (A.2) but the two body interaction is replaced by a truncated Lennard-Jones potential,

$$U(r) = \begin{cases} U_{\mathrm{LJ}}(r) - U_{\mathrm{LJ}}(r_{\mathrm{c}}) & \text{if } 0 \le r < r_{\mathrm{c}} \\ 0 & \text{otherwise,} \end{cases} \tag{3.6}$$

where $r_{\mathrm{c}} = 2.5\sigma$. The Lennard-Jones potential has the standard form,

$$U(r) = 4\epsilon \left[\left(\frac{r_m}{r} \right)^{12} - \left(\frac{r_m}{r} \right)^{6} \right], \tag{3.7}$$

where $r_m = 2^{-1/6} r_0$ and ϵ measures the potential-well depth (which is adjusted by the value at the truncation point). The selection of FENE and LJ functions matches exactly with those used in Ref. [146]. Reported by careful Monte Carlo studies utilizing the Wang-Landau algorithm [148, 146], it is well-known that the FENE model exhibits three different states in the low-energy region, globule, anti-Mackay (anti-Mackay), and Mackay (M). In the Wang-Landau algorithm, e is used as the system parameter, one could convert all languages to a low-temperature description but we stay here with the low-energy description for simplicity. The benefits of using e as the system parameter are discussed in Appendix A.3.2. With the decreasing of the energy, configurations of the FENE model will firstly exhibit the coil-to-globule transition, and then a further decreasing of the energy will trigger globule-to-anti-Mackay and anti-Mackay-to-Mackay transitions. These three structures have subtle structural differences that cannot be distinguished by direct visualization in Figs. 1.3(b), (c) and (d). In particular, the crystalline anti-Mackay and Mackay states differ delicately from each other by the way that monomers are stacked [Figs. 1.3(f) and (g)]. Similarly, 5×10^3 independent configurations are produced at every specified energy e. For FENE, transition points are determined by looking into specific-heat-like γ curve. Fig. 3.2(d) and Fig. 3.4 shown that in FENE model, Monte Carlo method gives the transition points of coil-to-globule, globule-to-anti-Mackay and anti-Mackay-to-Mackay at $e = -1.80$, $e = -4.40$, $e = -4.74$ respectively. The detailed information of the Monte Carlo procedure and specific-heat-like γ are documented in Appendix A.3.2.

Strictly speaking, phase transitions can only be defined in the thermodynamic limit, these transition points (produced in a finite-N system) have characteristic phase-transition properties of a finite system [147, 148, 146].

3.3 Training an FNN with polymer samples

3.3.1 Constructing an FNN

The graphic demonstration of the FNN we used in this chapter is shown in Fig. 2.2. The simulated configurations of a three dimensional polymer chain with N connected monomers are mathematically represented by $3N$ spatial coordinates. In order to perform the feed-forward step, we have two choices on how configurational information is fed into the input neurons. The simple, naive option is to convert our problem to the image recognition problem discussed earlier. The conversion can be executed by discretizing three dimensional space into N_{i} grids, and then assigning a state of "occupied" as value 1, or "unoccupied" as value 0 on top of a particular grid point. Then, this converted pattern would be used for training or testing. Alternatively, in this chapter, we implement an end-to-end supervised learning strategy, that is we classify polymer configurations by directly using $3N$ raw spatial coordinates as $N_{\mathrm{i}} = 3N$ input. Then, the neural network is trained to learn the relationship between the raw coordinates vectors and their corresponding the labels. In the hidden layer, in total $N_{\mathrm{h}} = 100$ neurons are used. In the output layer, the number of the output nodes, N_{o}, will be set to 2 or 3 depending on the problems to be studied.

3.3.2 Training

The way to train the FNN used is documented in section 2.2.2. During the training period, cross entropy is adopted as the cost function, and a dropout regularization with a 50% dropout rate is applied to the hidden layer to avoid overfitting in the determination of the weights and biases [158]. As a technical note, the dropout mentioned in the last paragraph is a technique aims at preventing a FNN from overfitting during training. In each epoch, this technique randomly selects some hidden neurons and ignores the weights that connect these neurons and neurons in the next layer. One can imagine that different smaller FNNs are trained in different epochs. As a result, the FNN gains the capability of generalization.

3.4 Coil-to-globule transition

Some questions immediately arise. Is a FNN able to distinguish different polymer states? If a FNN is capable of it, does it rely on the order parameters to classify these states? Does

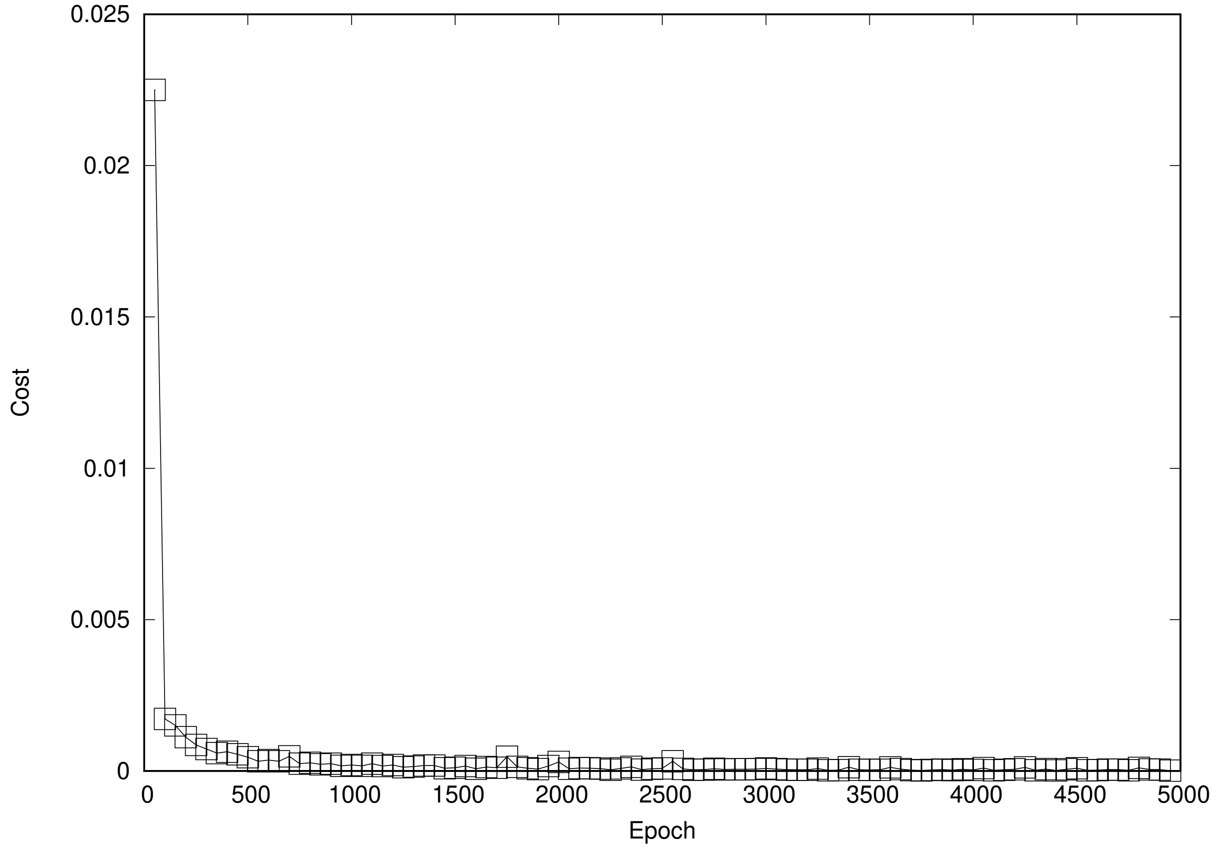

Figure 3.1: The cost with respect to epoch when training the FNN with samples containing coil and globe configurations.

the FNN trained on samples generated from one model can identify the samples generated from a different model? To answer these questions and in the mean time justifying the capability of the neural network in recognizing polymer states, we performed three numerical experiments based on a batch of sample that contains coil and globule state.

In the first experiment, 3×10^3 polymer configurations (specified by the coordinates of monomers after setting $a = 1$) at every specified temperature within the ranges $[0.5, 1.5]$ and $[2.5, 3.5]$ were labelled as training sets for the globule and coil states, respectively. The two normalized output neurons were designated as the coil and globule labels, which during training were assigned to have values $\nu = 1$ for the corresponding state, and $\nu = 0$ otherwise. No other information or estimators from Monte Carlo simulations, such as the temperature, reduced specific heat, or radius of gyration were used in the training.

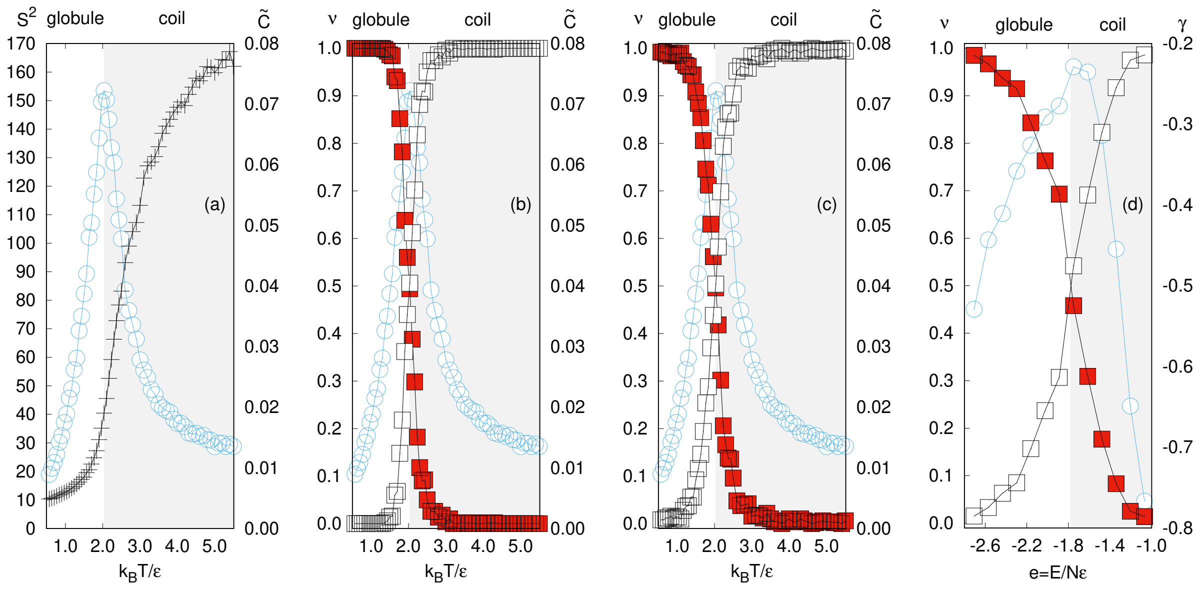

Figure 3.2: (a) Reduced mean-square radius of gyration (plus symbols, to the left scale) and specific heat (circles, to the right scale) as functions of the reduced temperature $k_\mathrm{B}T/\epsilon$, determined from the Monte Carlo simulations of GSM for $N = 102$, (b) the neural network outputs of test-recognizing independent GSM configurations, (c) the neural network outputs of test-recognizing independent, *normalized* GSM configurations, and (d) the neural network outputs of test-recognizing normalized FENE configurations. The filled and open squares represent the mean ν-values output from the globular and coil neurons, respectively. The circles in plot (d) represent the specific-heat-like γ (to the right scale) determined for the FENE model from a Wang-Landau Monte Carlo simulation. Error bars associated with all squares are smaller than the symbol size.

Fig. 3.1 shows the change of cost function with respect to training epochs. One can straightforwardly see that the cost is stable and small enough after around 5000 epochs. This is the indication of the FNN is trained.

Once the neural network was adequately trained and all neural network parameters were stable, we input 500 new configurations at every temperature between the range $[0.5, 5.5]$, which were not used in the training session, as the testing set. The averaged test values of the two output neurons, ν, are plotted in Fig 3.2(b) forming two curves behind the filled and open squares. It's straightforward to see that in the low temperature regime, only the neuron designated as globule labels output a number near 1, and in high temperature regime, only the neuron designated as coil labels output a number near 1, which demonstrated that neural network is capable of identifying coil and globule. More importantly, the intersection of two curves is interpreted as the transition point, since this is where neural network think the test data neither like coil nor like globule state. As Fig 3.2(b) shows, these curves cross with each other at $k_BT/\epsilon = 2.03$, identifying a coil-to-globule transition that in good agreement with the location of the $\tilde{C}$-peak, regardless of the fact that the neural network model was trained in temperature ranges farther away from the transition point.

The coil and globule states have distinguishably different shape, represented by the radius of gyration, S^2. To show that the neural network is not simply recognizing coil and globule from their different overall sizes, in our second experiment we normalized all coordinates, of the training and testing sets, by a factor $1/S$. On average, the polymer configurations now have the same normalized radius of gyration ($= 1$), across the entire studied temperature range. The network was then trained in a similar manner described above, with the normalized coordinates. The quality of output neurons to indicate the coil and globule states for the testing data is equally good as in the previous case, shown in Fig. 3.2(c).

In the third experiment, we tried to test on the transferability of a trained neural network. While these numerical experiments were performed by using the configurational data generated from GSM, we placed the network into the ultimate test in the third numerical experiment. This time, the neural network parameters determined in the second experiment were retained and we asked the network to recognize the configurations generated from the FENE model, in which completely different potential functions were used than in the square-well GSM. Furthermore, instead of producing the configurations from the canonical ensemble where k_BT/ϵ is used as the system parameter, we generated configurations from the Wang-Landau algorithm for microcanonical ensembles, in which the total (reduced) energy per particle $e = E/N\epsilon$ is directly used as the system parameter. At each e-value illustrated in Fig. 3.2(d), 1×10^3 independent FENE configurations, normalized by their corresponding S, were sent to the GSM-trained network for testing. The two recognition curves produced from the output neurons, shown in the figure as the underlying curves behind filled and open squares, predict a coil-to-globule transition point at $e = -1.8$. This prediction can be independently verified by examining the $\gamma(e)$ curve, which is a specific-heat-type measurement in reduced units defined in the microcanonical ensemble [146]. The data represented by circles in Fig. 3.2(d) was calculated based on the density of states determined from the Wang-Landau algorithm; it shows a peak at the same location as the one successfully predicted by GSM-trained neural network.

3.5 Low-energy polymer states

The numerical experiments in Sec. 3.4 ascertains that the FNN has no problem identifying different polymer states. With the intention of exploring the upper limit of this capability, in this section, as a final numerical experiment, we are trying to challenge the neural network to recognize three very similar states, using three neuron nodes in the output layer, each assigned to recognize globule, anti-Mackay and Mackay separately.

The network was trained with FENE configurations in the energy range $e = [-4.3, -4.16]$, $[-4.7, -4.5]$, and $[-4.9, -4.8]$, where the globule, anti-Mackay and Mackay structures, respectively, can be clearly labelled. The training data contained 3×10^3 configurations at every energy bin. Fig. 3.3 shows the change of cost function with respect to training epochs. One can straightforwardly see that the cost is stable and small enough after around 10000 epochs. After the network was adequately trained, it was tasked to recognize an independent set of test data covering the entire energy range in Fig 3.4. Over 10^3 configuration samples were used at every energy bin for this purpose. The mean ν-values of the test output, from the globule, anti-Mackay, and Mackay nodes, are represented in Fig 3.4 by filled squares, open diamonds, and filled diamonds. The intersections of the interpolated curves predict that globule-to-anti-Mackay and anti-Mackay-to-Mackay transitions take place at $e = -4.40$ and $e = -4.74$, respectively. These neural network predicted transition points can be confirmed by an examination of the specific-heat-like γ function, independently calculated from the Wang-Landau Monte Carlo simulations. From the γ-peaks, indicated by the blue circles in the plot, we determine that the globule-to-anti-Mackay and anti-Mackay-to-Mackay transition points are at $e = -4.40 \pm 0.03$ and $e = -4.74 \pm 0.03$ respectively, which agree well with those from the neural network predictions. Albeit the identification of these states, in many cases, is arduous for human, or require exquisitely designed order parameters, the results of this experiment show that FNN can identify both states and transition points based on suitable amount of training.

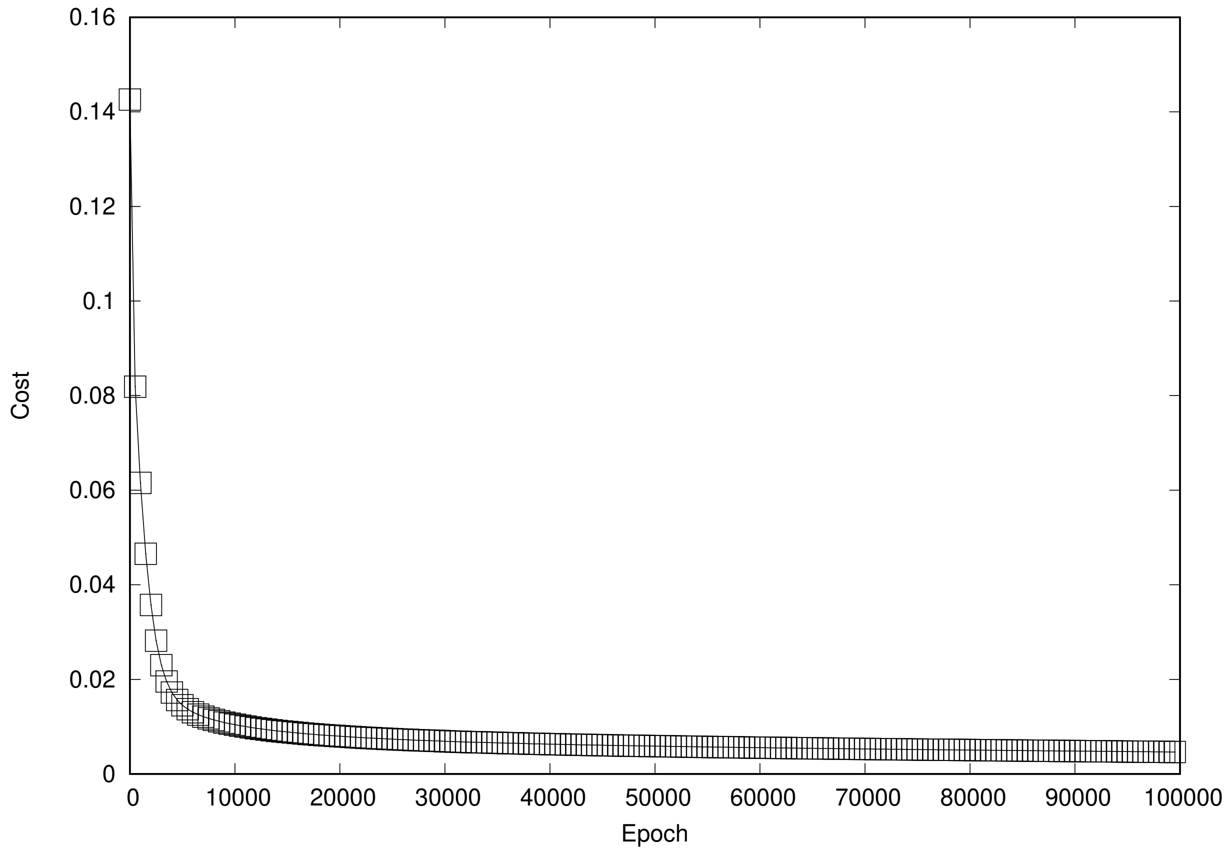

Figure 3.3: The cost with respect to epoch when training the FNN with samples containing globe, anti-Mackay, and Mackay configurations.

3.6 In search of a phase transition

The above numerical experiments demonstrated the FNN's versatility in recognizing polymer configurations and the usefulness of neural network in determining the transition points. There are two essential questions we have not adequately addressed. 1) How does the neural network predicted location of the transition depend on the range of used training data? 2) Would the network mistakenly identify a phase transition for a system where a certain physical property smoothly crosses over from relatively large to small values without going through a real phase transition?

To answer the first question we return to the GSM Monte Carlo data over a wide range of the reduced temperature, $[0.5, 3.5]$. We conducted a series of 18 independent

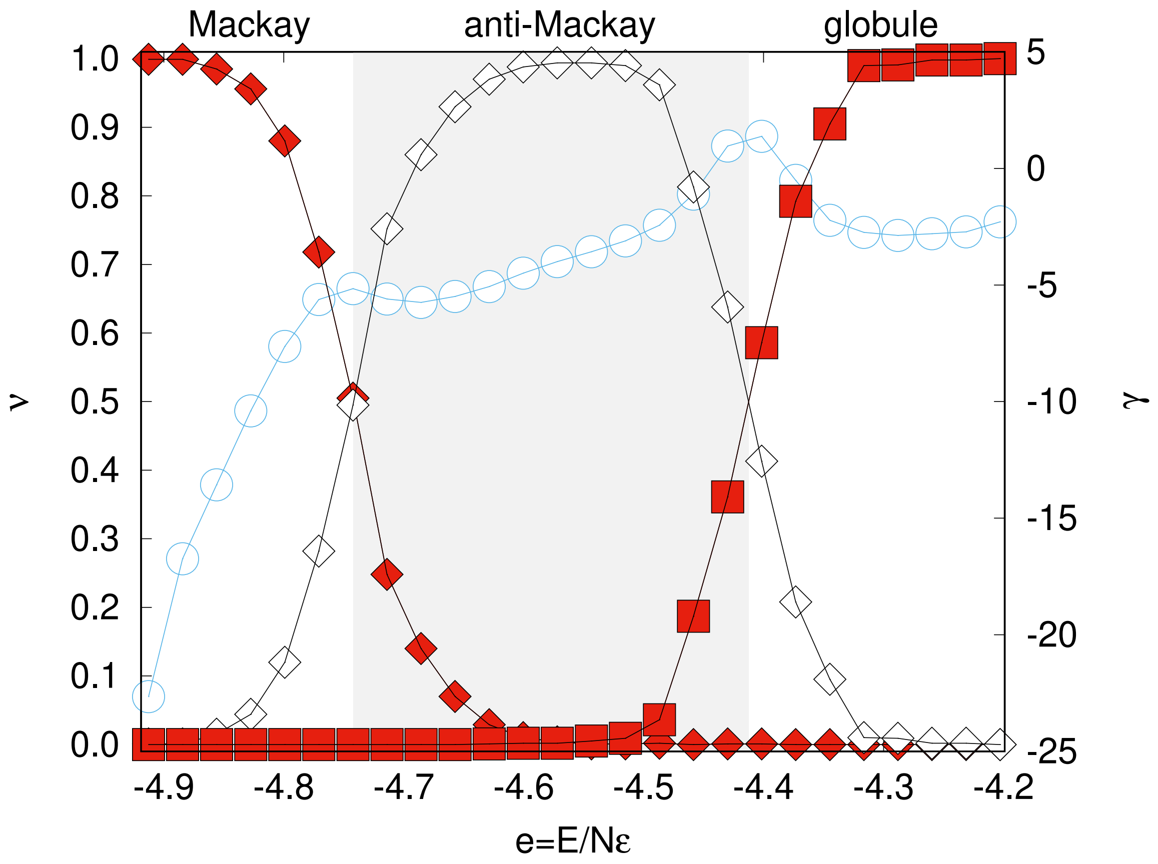

Figure 3.4: Mean neural network outputs ν (square for globule, open diamonds for anti-Mackay, and filled diamonds for Mackay) from the test samples, after the network is trained to recognizing these states in regimes where they are stable. In the background, the reduced specific heat-like γ (circles, to the right scale) was independently produced from the Monte Carlo simulations. Error bars are smaller than the symbol size.

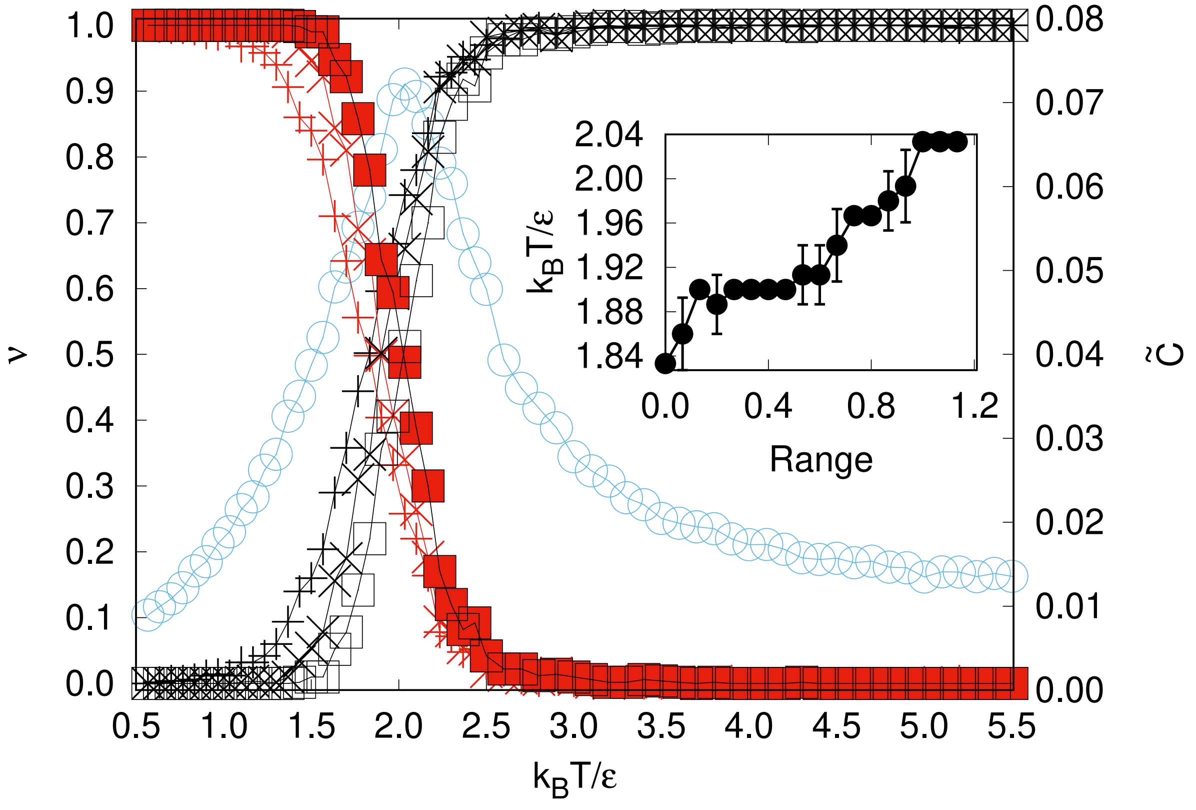

Figure 3.5: Average neural network output ν from the globule and coil neurons on the testing configurations for neural network models trained in various temperature ranges. The inset in plot is the neural network predicted transition temperature as a function of the training temperature-range used. Error bars are smaller than the symbol size, unless otherwise plotted. The plus, cross, and square symbols represent the mean ν from the two output neurons on test samples, of neural network models initially trained in the temperature range, Range $= 0.00, 0.53$, and 1.20, respectively. The blue circles represent the same data in Fig. 3.2(a).

training sessions, each assigned a different training temperature range, away from the coil-to-globule transition point. The first session treated Monte Carlo configurations produced from the system at two temperature values, one on the extreme left and the other extreme right, where the coil-to-globule transition sits in the middle. In the subsequent sessions the training temperature ranges expanded from the extreme left to the center and extreme

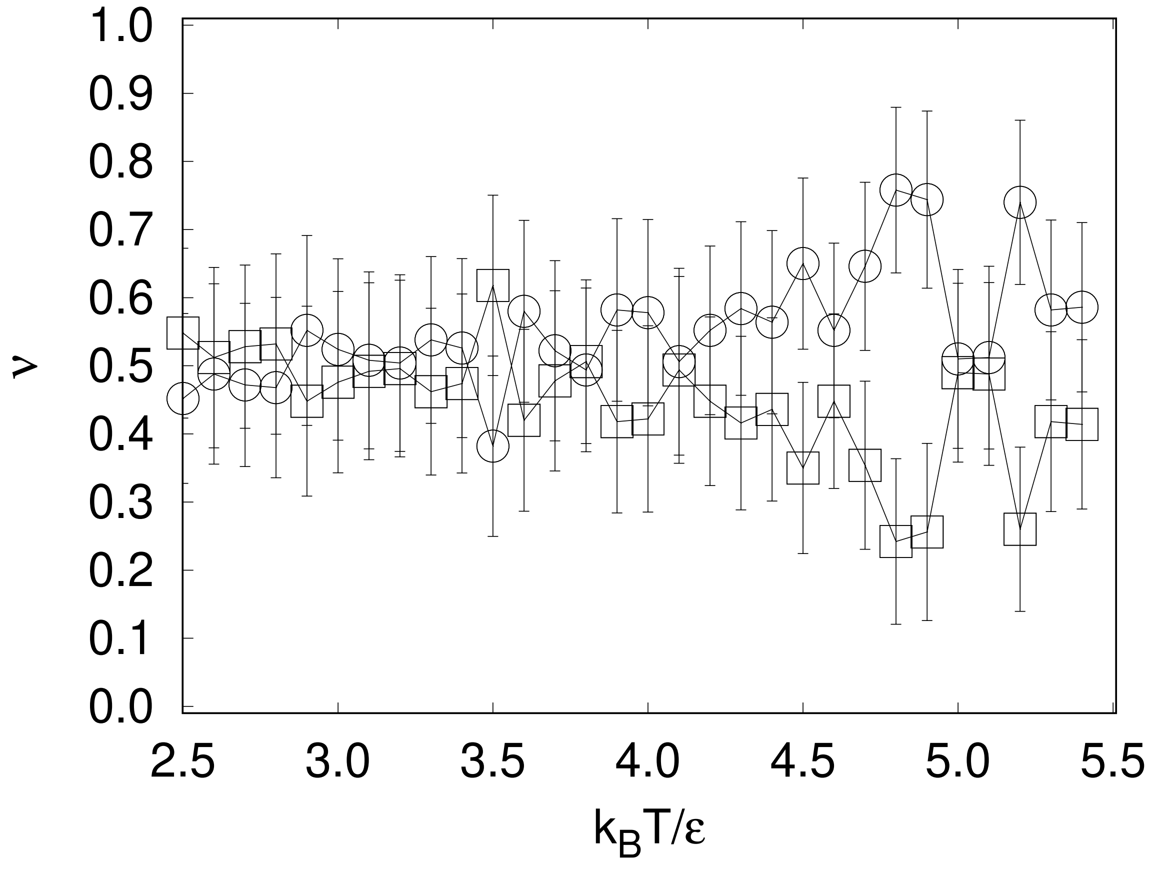

Figure 3.6: Average neural network output ν from "phase-1"- (squares) and "phase-2"-nodes (circles) on the test configurations in the temperature range $[2.5, 5.5]$ for neural network models trained by using low- and high-temperature samples. This temperature range contains no actual phase-transition point.

right to the center, incrementally. These trained networks were then put into test, by inputting independent configurations over the extended $[0.5, 5.5]$-range. The mean output ν-values from the C and globule nodes are illustrated in Fig 3.5 for a few selected sessions. As the training range expands the FNN predicted transition temperature converges to a fixed point; the final converging temperature agrees with the one determined by the Monte Carlo $\tilde{C}$-peak, 2.03. This study suggests that the approximate location of the transition temperature can be already estimated by using early training ranges far away from the transition point and that the more precise determination of the transition point can be achieved by progressively adding configurations closer to the transition point. The procedure actually reveals a mechanism of finding a phase transition point, without *a priori* knowledge of its existence, by taking two small training ranges as the starting point and proposing a phase transition point somewhere in between.

To answer the second question, we conduct a numerical experiment on a neural network with GSM Monte Carlo data in the reduced temperature range $[2.5, 5.5]$. Within this coil region, both S^2 and average energy (not shown) have significant variations. We enforcedly train the neural network so that the two output neuron nodes mistakenly regard configurations in the range $[2.5, 3.5]$ as in phase-1 and $[4.5, 5.5]$ as in phase-2. We then test the network with independent configurations over the entire $[2.5, 5.5]$ range. The results of the output nodes are plotted in Fig. 3.6. Each node vaguely recognizes the configurations as "phase-1" or "phase-2", with a mean ν-value hovering around 0.5 in high uncertainties. No clear signals, such as those determined above for true phase transitions, exist.

3.7 The order of a phase transition

In the previous sections, we illustrated the method of using FNN to study phase transitions, including identifying different polymer states and locating the transition points. In short, this method investigates the average of the FNN output on test samples. Based on the test samples generated from different temperature or energy, we can eventually obtain 2 to 3 ν curves based on the problem to be studied. One looks into the intersect of two $\nu(e)$ curves, and interpret this intersect as the transition point. Let's call this method "50% rule".

In comparison to the 50% rule, recently, we discovered the transition points could be located based on a different philosophy. We are aiming at studying the standard deviation behaviour of the FNN outputs in each energy bins. Notice the fact that the samples are generated by the Monte Carlo method. For the energy bins that are very far from the transition point, most of the configurations sampled belong to one same state. On the contrary, the configurations generated near the transition point could be a mixture of both states. When an FNN makes inference on test samples, if most of the samples belong to one state, then the FNN tends to make the same inference, and the standard deviation of the outputs of each node will be small. Otherwise, if the samples consist of a mixture of both states, the FNN is expected to make different inferences. The standard deviation of the outputs of each node will be large. Therefore, if the standard deviation of the FNN

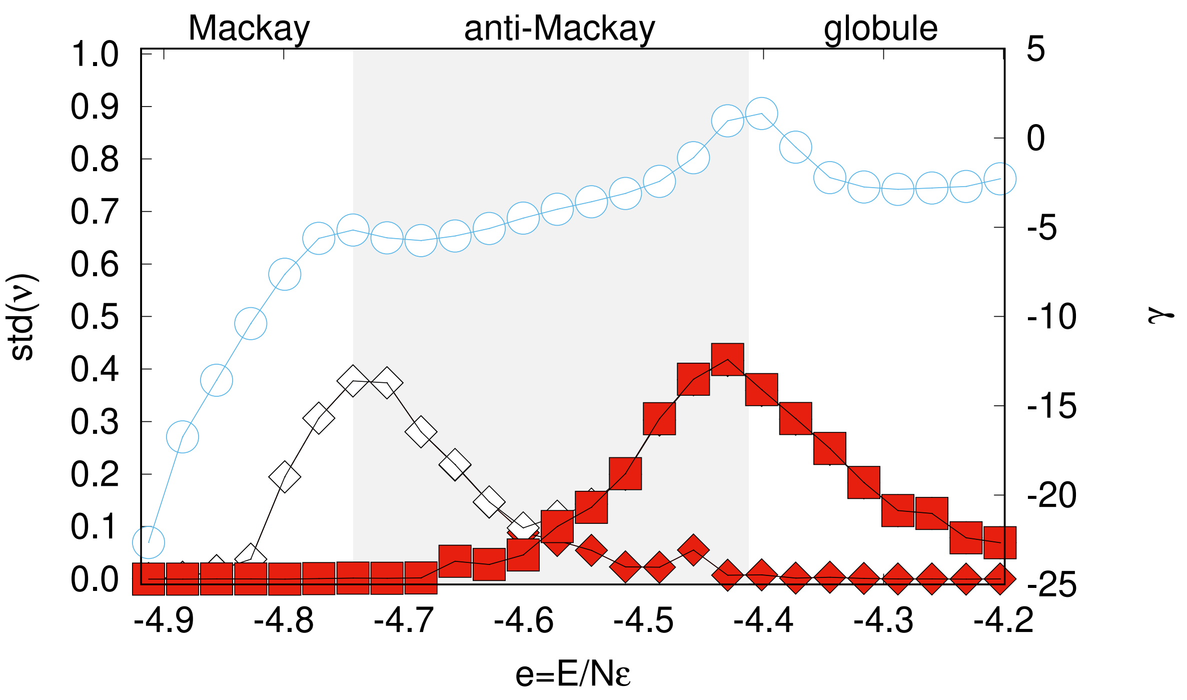

Figure 3.7: Standard deviation of neural network outputs ν (square for globule, open diamonds for anti-Mackay, and filled diamonds for Mackay) from the test samples, after the network is trained to recognizing these states in regimes where they are stable. In the background, the reduced specific heat-like γ (circles, to the right scale) was independently produced from the Monte Carlo simulations. Error bars are smaller than the symbol size.

outputs calculated from different energies or temperatures are plotted together, the curve we obtain should peak at the transition point.

In this section, we chose to study the data set that contains globule, anti-Mackay and Mackay configurations. It is well understood that the globule to anti-Mackay transition is a first order transition and the anti-Mackay to Mackay transition is a second order transition [146], which makes it a good example data set to explore if analyzing the standard deviation of FNN outputs can identify the structures or identify the order of the transition. We use the same data set and FNN architecture as in Sec. 3.5. Similarly, the network was trained with configurations sampled from the FENE model in energy ranges $e = [-4.3, -4.16]$, $[-4.7, -4.5]$, and $[-4.9, -4.8]$ which contains globule, anti-Mackay and Mackay structures, respectively. However, after the FNN is adequately trained, the standard deviation, instead of the mean, of the test outputs at every energy bins are calculated.

The results for globule, anti-Mackay, and Mackay nodes are represented in Fig. 3.7 by filled squares, open diamonds, and filled diamonds. In this method, the peaks of the interpolated ν-curve determine the transition points. By comparing Fig. 3.4 and Fig. 3.7, we find the transition points given by standard deviation of ν agree well with the transition points given by the 50% rule and specific-heat-like γ curve.

Furthermore, we visualized the distribution of the FNN outputs, given a set of test samples near transition points. Fig. 3.8 and Fig. 3.9 show the distributions of FNN outputs near the globule-to-anti-Mackay transition and the anti-Mackay-to-Mackay transition respectively. We expect the distributions of these FNN outputs behave differently at these transition points so that we can identify the order of these transitions. However, the distributions in Fig. 3.8 and Fig. 3.9 look the same to the author, which means we may need to consulting other methods to determine the order of a phase transition. At the moment, we haven't found a way to determine the order of a transition.

3.8 Summary

In this chapter, we described the training of FNN to recognize diversely and subtly different polymer states produced from Monte Carlo simulations. The example used here is a classical molecular system displaying gas-, liquid-, and crystal-like structures at various energies.

We demonstrated that the neural network can classify all the structures observed, such as coil, globule, anti-Mackay and Mackay, and identify all the transitions observed. The direct use of molecular coordinates as input into the neural network underlies the robustness and simplicity of our approach, and suggests that other simulation tools, such as molecular dynamics, could be used to produce configuration data for supervised learning as well. The outcome of this work provides a compelling reason to incorporate machine learning techniques into molecular simulations more generally, as a powerful hybridized computational tool for the future study of other polymeric systems.

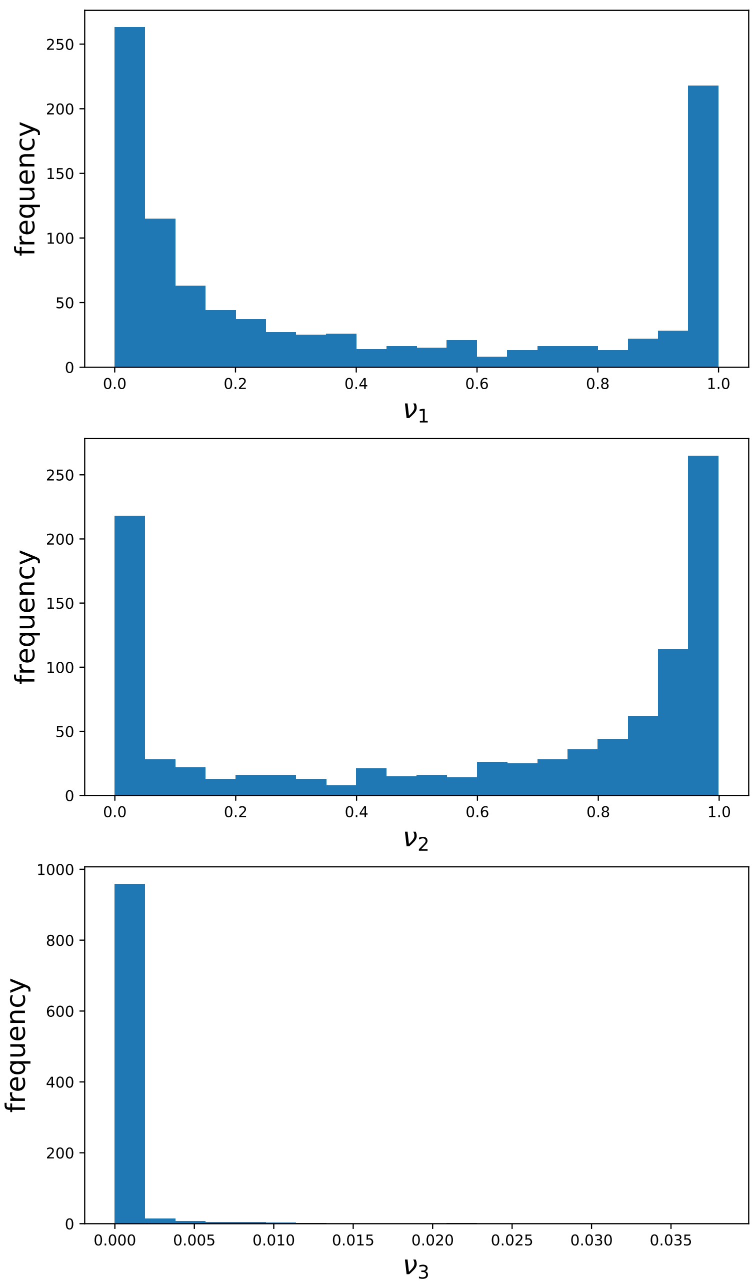

Figure 3.8: Histogram of the FNN outputs calculated from the samples generated near globule-to-anti-Mackay transition. ν_1, ν_2 and ν_3 represent the nodes designed to identify globule, anti-Mackay and Mackay respectively.

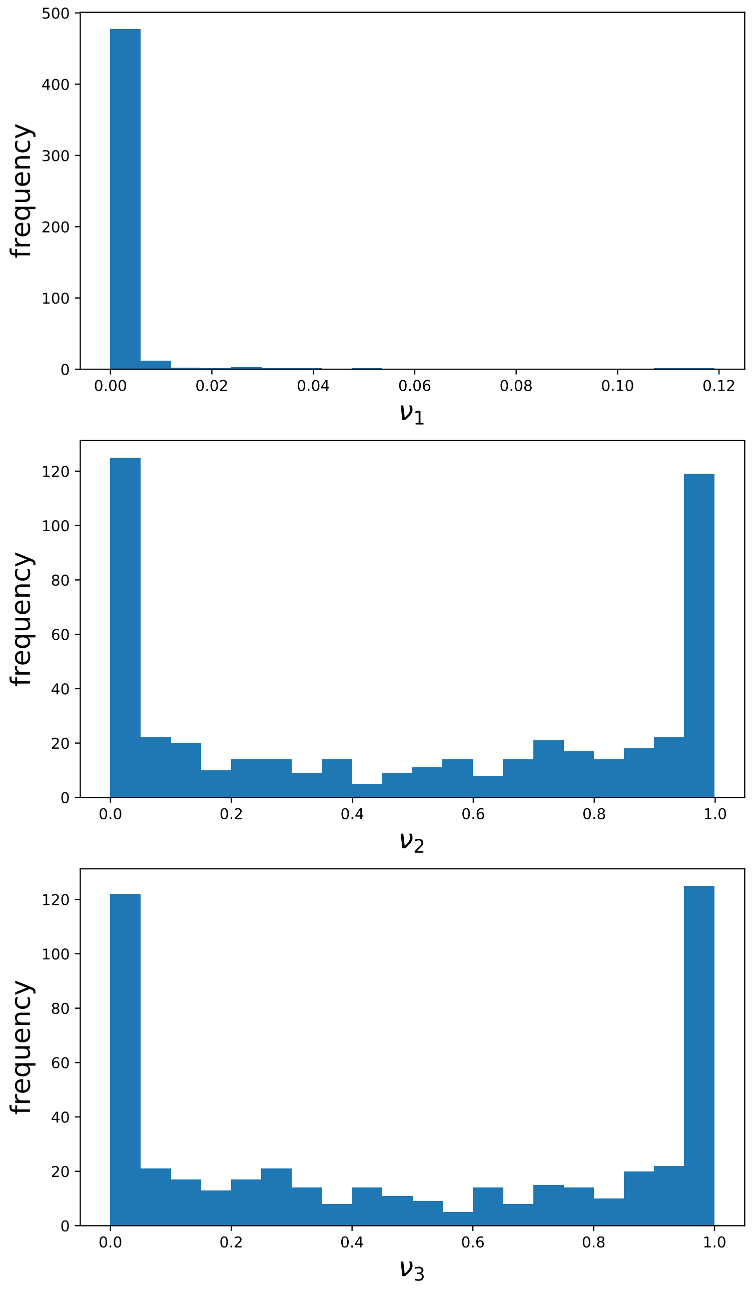

Figure 3.9: Histogram of the FNN outputs calculated from the samples generated near anti-Mackay-to-Mackay transition. ν_1, ν_2 and ν_3 represents the nodes designed to identify globule, anti-Mackay and Mackay respectively.

Chapter 4

Identifying polymer states by unsupervised learning

4.1 Introduction

Taking advantage of the formidable capacity of machine learning methods in terms of classification, dimensionality reduction and cluster analysis, many compelling studies have been devoted to classify the phase of matter and identify phase transition point [15, 19, 26, 22, 70, 101, 143, 129, 124, 182, 196, 195, 172, 23, 175, 184, 136, 194]. Basically, based on if the training data have labels or not, these applications can be categorized into two types, i.e. supervised learning and unsupervised learning. The studies [101, 195, 143, 22, 129, 194, 15, 124, 182, 19, 196] using the former one indicate that the machine learning models are trained by data associated with the correct labels before prediction. Whereas, in Ref. [23, 26, 70, 136, 172, 175, 184], using the unsupervised learning techniques, data without labels are divided into several groups through cluster analysis and simultaneously features standing for a low-dimensional representation of data are extracted. The application of unsupervised learning on studying physics starts from Wang, who utilizes PCA to classify states sampled from the Ising model [175]. He shows that PCA can reduce the dimensionality of raw spin configurations from the number of spins to one. He then finds that the clustering behaviour of low-dimensional representations is highly determined by the order parameter. Hu et al. [70] have examined the performance of PCA on classifying the states sample from more complicated spin models. They find PCA is able to classify the states that do not have clear order parameter, such as the states sampled from anti-ferromagnetic Ising model on triangular-lattice. They also discover that

PCA have limited capability. For identifying states in more complicated spin models, e.g. XY model, one need to use advanced methods [184]. Compared with supervised learning, unsupervised learning methods have the advantage of avoiding the time-consuming data labels and playing the role of a preprocessor to data in supervised learning. Recently, a hybrid approach combining both techniques is developed as well for identifying the phase transition[170, 108]. Most of these applications mainly concentrate on the spin models in condensed matter physics, such as Ising model [19, 129, 175, 184, 70, 170, 108], Hubbard model [22, 15, 23, 26], XY model [172, 184, 70], and Anderson model [124, 26]. Nevertheless, applications in the field of soft matter largely lagged behind conventional condensed matter counterparts.

In this chapter, we explore the application of the unsupervised learning methods in polymer physics, especially on the structural transition of polymer configurations. The unsupervised learning is commonly adopted in the dimension reduction to search for the lower-dimensional feature space (i.e. underlying manifold) in which the raw data can be embedded suitably. The most attractive advantage is its ability in the clustering analysis without correct data labels in advance. Herein, we implement the unsupervised learning scheme to extract the characteristic quantities that can distinguish between different structures from polymer configurations in terms of few types of dimensionality reduction techniques, and moreover detect the location of phase transition. Our study exhibits the great power of unsupervised learning scheme on the aspects of the recognition of structural transitions and the inference of underlying feature for the polymeric systems, in face of the challenge of big data with complex structures.

4.2 Main procedure

In this chapter, the polymer model we studied is FENE, which has exactly interactions and parameters as the FENE model documented in chapter 3. The Monte Carlo methods we used to generate samples are the same as the one documented in appendix A.3.2. The reason of these choices is to make sure the configurations studied in this chapter contain the same states and transition points as the configurations studied in chapter 3, so that we can compare results produced form the unsupervised learning methods in this chapter and supervised learning methods in chapter 3.

Once the samples are generated, they are organized into a data matrix, **X**, in a way described in section 2.3. When PCA is applied to **X**, one can follow Eq. (2.6) and Eq. (2.7) to obtain all the eigenvectors and normalized eigenvalues of the correlation matrix. Then, depending on the number of dominant principal components, following Eq. (2.8), **X** can be converted into its low dimensional representation, **Z**.

4.3 Coil-to-globule transition

As we know, for the FENE model, the coil-to-globule transition takes place in the high energy regime, corresponding to a pronounced peak of the specific-heat-like γ [146]. We prepare 400 configurations by the Wang-Landau method at each energy bin equally spanned in the energy range $e = E/N\epsilon \in [-3, 1]$, within which the system undergoes a coil-to-globule transition [182].

In the following section, we will use PCA to distinguish coil and globule states. As a linear dimension reduction technique, PCA [128, 78] is an unsupervised learning method widely utilized to extract the feature of raw data. The raw data, via PCA, are embedded in a lower-dimensional space in which the largest variations of data are captured. Equivalently, one tries to explore the lower-dimensional representation of the original data, in order to minimize the average reconstruction error.

Fig. 4.1 shows the first 20 normalized eigenvalues. As shown in Fig. 4.1, there is a jump between the third and the fourth eigenvalue. Apparently, there are three predominant components in the leading eigenvalues, which indicates that polymer configurations can be reduced and projected to a three dimensional space spanned by W_1, W_2 and W_3. In the coordinate system formed by the corresponding three orthogonal directions, as shown in Fig. 4.2, all configurations are roughly divided into two parts through the visualization of two-dimensional projections, in accordance to the variation of energy e, we observed the configurations that have lower energy tends to cluster inside while the configurations that have higher energy tends to spread around center.

It reveals that the configurations in the central regime (low energy) have a much smaller radius of gyration (R_g) than the ones in the outside regime (large energy), this hints that PCA may have discovered the importance of R_g from different configurations.To make a more clear visualization of the relationship between R_g and the 3D representation of configurational data, we colour codes the points in the three dimensional space spanned by W_1,W_2 and W_3 by its corresponding R_g instead of e, as shown in Fig. 4.3. It is clear that the distinguishable feature for coil and globule states discovered by PCA is R_g, which directly reflects the volume occupied by polymers. Motivated by the projections approximately exhibiting the distribution with a circular shape as shown in Fig. 4.3, we intuitively propose to attain one reduced dimensionality parameter by the non-linear transformation

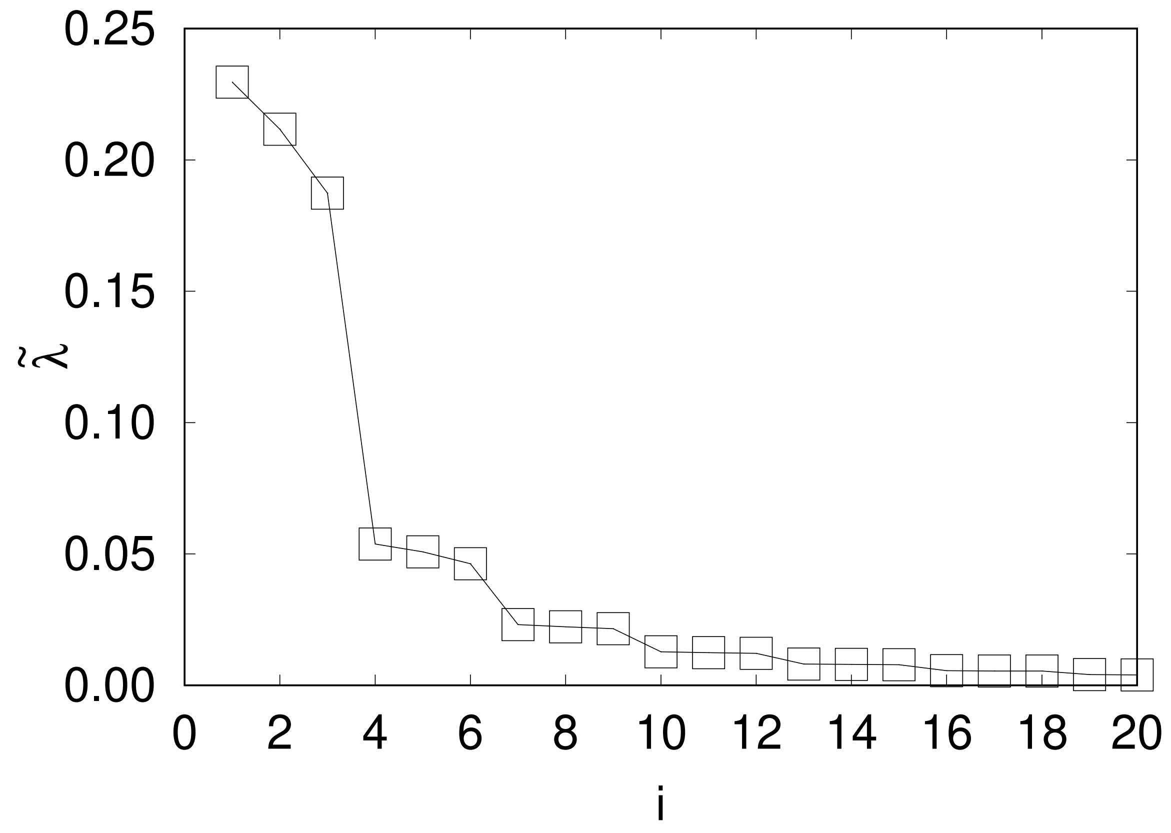

Figure 4.1: The top eigenvalues $\tilde{\lambda}_i$ obtained from PCA for coil and globule states in the energy range $e \in [-3, 1]$.

of first few coordinates of data in the low dimensional representation according to $\sum_{l=1}^{K} Z_l^2$. To analyze the relationship between $\sum_{l=1}^{K} Z_l^2$ and R_g, we proposed a correlation analysis by computing the Pearson correlation coefficient $\rho(\sum_{i=1}^{K} Z_l^2, R_g^2/\sigma^2)$ which measures the linear dependence between two variables. One finds that the reduced dimensionality variable $\sum_{l=1}^{K} Z_l^2$ shows a strong linear dependence on the $(R_g/\sigma)^2$, as illustrated in Fig. 4.4. This dependency further justified that PCA learned and distinguished coil and globule states by discovering the R_g of each configuration.

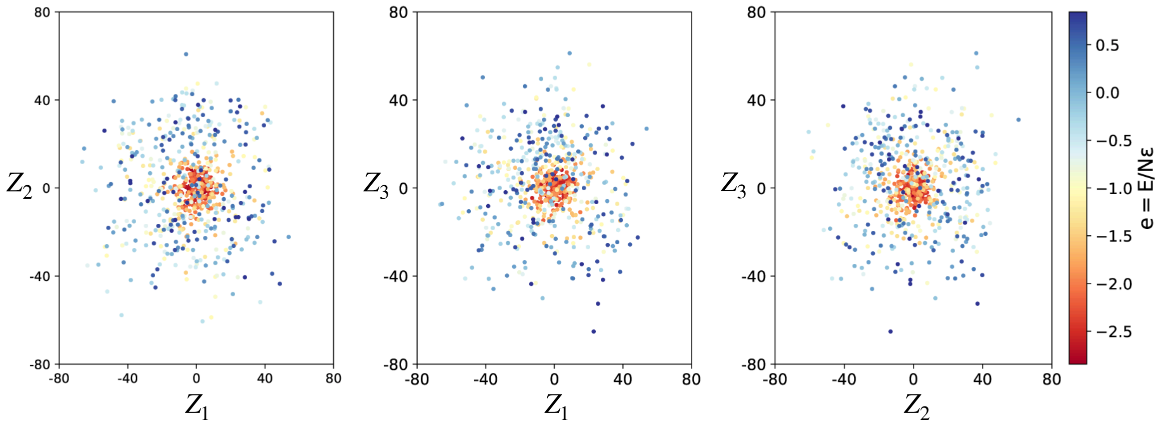

Figure 4.2: A visualization of coil and globule configurational data, colored according to e, in the energy range $e \in [-3, 1]$. The data is firstly projected to the space spanned by top three eigenvectors W_1, W_2, W_3 obtained from PCA, in order to obtain a 3D representation of data. Z_1, Z_2, Z_3 are the coordinates of the data in this representation. The colored dots in these plots, from left to right, are the two-dimensional projections of the 3D data to the W_1OW_2, W_1OW_3, W_2OW_3 plane respectively.

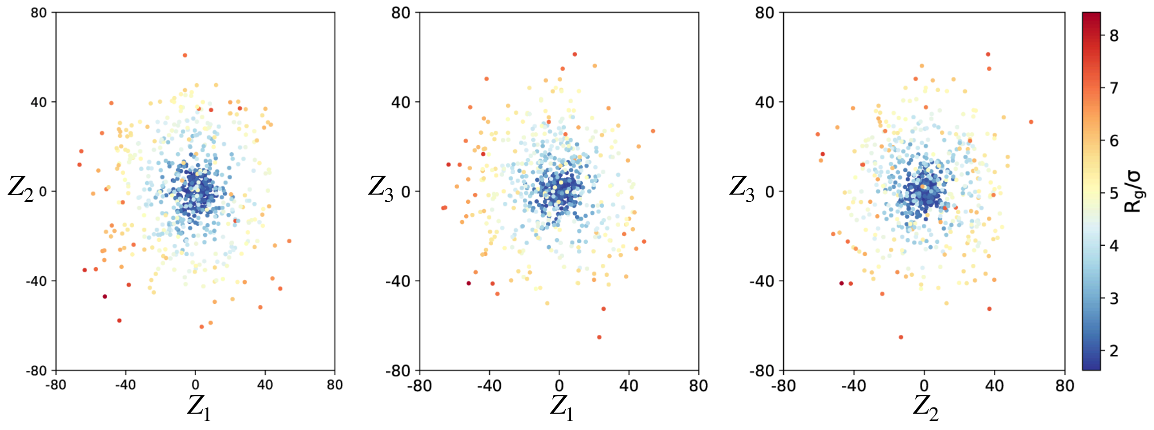

Figure 4.3: A visualization of coil and globule configurational data, colored according to R_g/σ in the same range of energy as Fig. 4.2. The colored dots in these plots, from left to right, are the two-dimensional projections of the same 3D data as in Fig. 4.2 to the W_1OW_2, W_1OW_3, W_2OW_3 plane respectively.

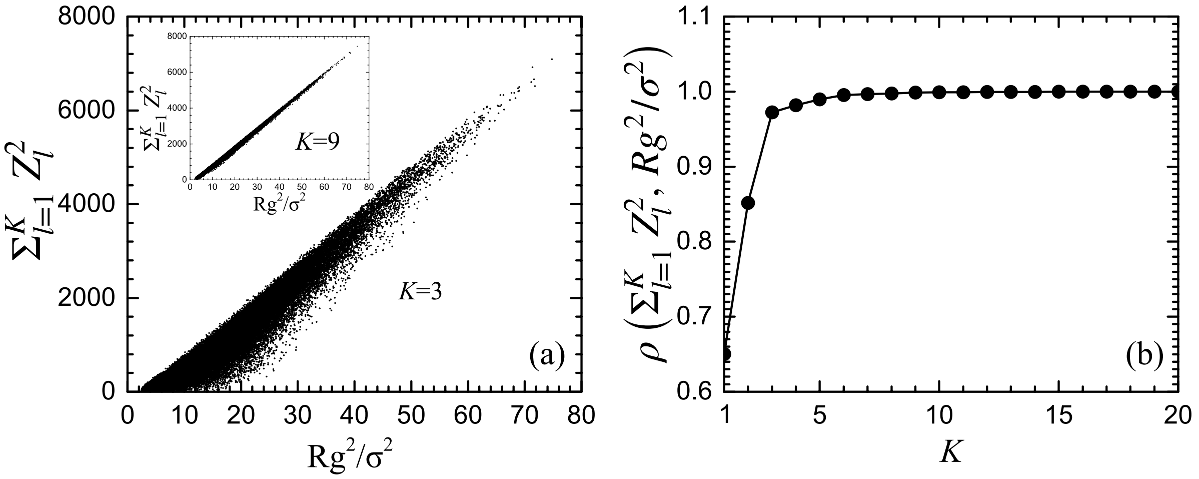

Figure 4.4: The behaviours of the nonlinear transformation of the top K eigenvectors are described according to $\sum_{i=1}^{K} Z_l^2$. (a) The mapping between $\sum_{i=1}^{K} Z_l^2$ ($K = 3$) and R_g^2/σ^2. The insert plot is for $K = 9$. (b) The Pearson correlation coefficient $\rho(\sum_{i=1}^{K} Z_l^2, R_g^2/\sigma^2)$ as a function of K. R_g is the radius of gyration for polymers.

4.4 Globule-to-anti-Mackay-to-Mackay transition

With further decreasing the energy, using FENE model, as we know, one can obtain three states globule (G), anti-Mackay (aM) and Mackay (M) in the low-energy region parameterized by the e, through adopting Monte Carlo simulation based on the Wang-Landau algorithm [148, 146]. Following the same approach, we perform PCA on a stack of configurations sampled from energy regime $e = [-5, -4]$ by the Wang-Landau method, which consist of globule, anti-Mackay, and Mackay structure. Similarly, this energy range is sliced into 30 energy bins, 400 configurations are drawn for each energy bin. After PCA is applied, Fig. 4.5 shows the normalized eigenvalues. Compared with Fig. 4.1 where the existence of a jump easily reveals the dominant dimension as three, the eigenvalues shown in Fig. 4.5 decay smoothly. From the inset, we can see that the first eigenvalue is not significantly large than the 20th eigenvalue, which means in order to find low dimensional representation for the mixed globule, anti-Mackay and Mackay samples, we need a space spanned by at least 20 eigenvectors. Equivalently, PCA can only reduce the original configurations to a space spanned by at least 20 W_is. Unfortunately, it is unrealistic to find a proper way to visualize this 20 dimensional data to perform cluster analysis, which demonstrates that the subtle structural difference among the three states is hard to be distinguished by a

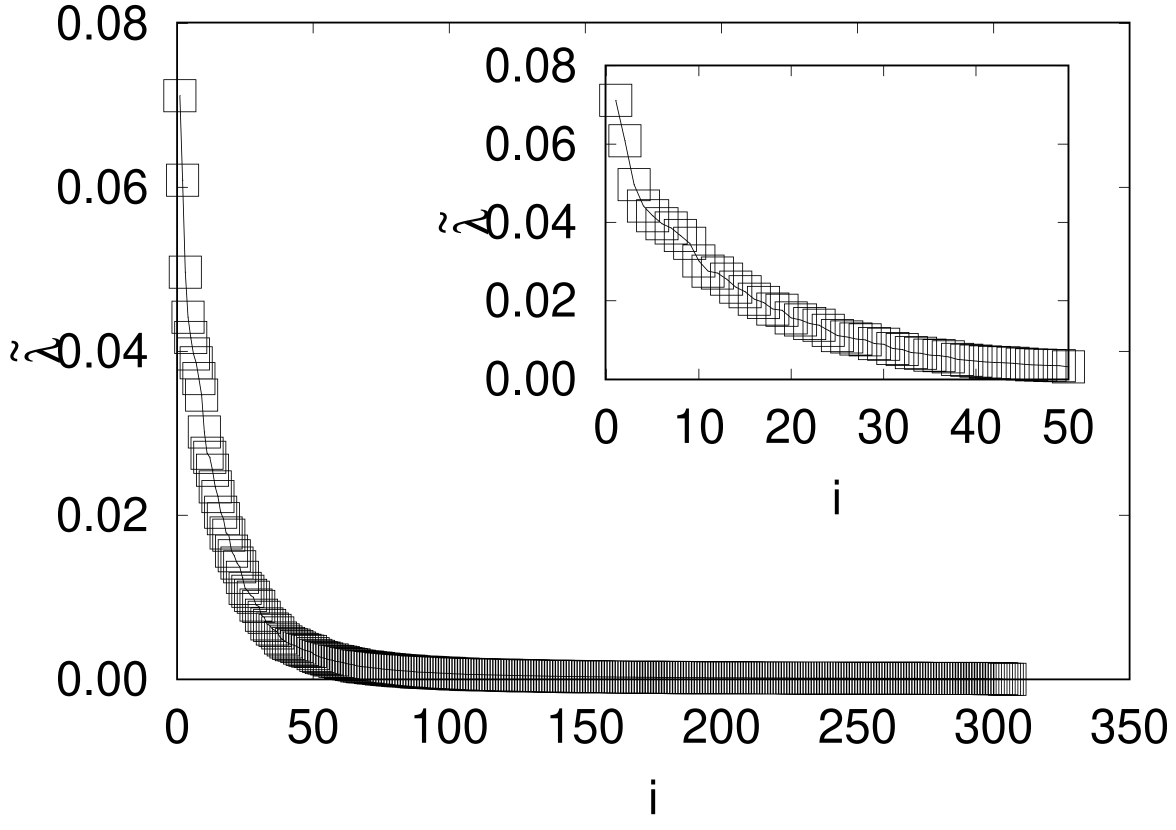

Figure 4.5: All normalized eigenvalues λ_l obtained from principal component analysis (PCA) for globule, anti-Mackay, Mackay states in the energy range $e \in [-5, -4]$. The inset shows the first 50 normalized eigenvalues.

simple visualization through PCA, especially for aM and M states differ each other slightly in the stacking manner of particles at the outer layer. As we saw from the last section, PCA distinguishes different states based on R_g. Because of the collapsed states are similar in R_g, that is the reason why PCA is unable of distinguishing these collapsed states.

To distinguish collapsed states by unsupervised learning, we need to turn to more powerful methods. Xu et al. [189] purpose to use diffusion map, which is a nonlinear dimensionality reduction technique firstly introduced by Coifman and coworkers [24], as the dimensionality reduction method. They have shown that the diffusion map is able to reduce the dimensionality of samples contains globule, anti-Mackay and Mackay to 3. The correlation analyses find that the diffusion map discovers Q_6^{core} as the parameter to distinguish the difference between globule and anti-Mackay states and Q_6 as the parameter to distinguish of the difference between anti-Mackay and Mackay states.

4.5 Identifying transition points

Although unsupervised learning methods are able to distinguish multiple polymer states, the identification of transition temperature or energy between structures can only be roughly estimated by analyzing the low dimensional representations, such as looking for "gaps" between clusters [175], or resorting to some particular clustering algorithms like K-means [73], k-nearest neighbors [155, 29], support vector machines [14, 25]. These methods usually can not make accurate predictions about the transition points.

Recently, van Nieuwenburg et al. developed an elegant hybrid approach that combines the unsupervised learning techniques and neural network to explore the phase transition in several condensed-matter systems[170]. Through training the neural network with states labelled based on the hypothetical transition point, which might be deliberately designated incorrectly, one can obtain the corresponding accuracy of the prediction. When the hypothetical transition equals the actual transition point, the states are correctly labelled, and the correct label will lead to higher accuracy. Eventually, the determination of a transition point is equivalent to look for the turning point at which the data are correctly labelled. For convenience, herein, we call this approach the W-shape scheme, on account of the W-like profile of predictive performance. One fascinating trait of the W-shape scheme is that it is essentially an unsupervised learning approach, owing to the training and test data that do not need *a prior* correct label.

In the following part, we will demonstrate the utilization of the W-shape scheme in our polymeric system. Assuming that there are two states A, B to be classified. The

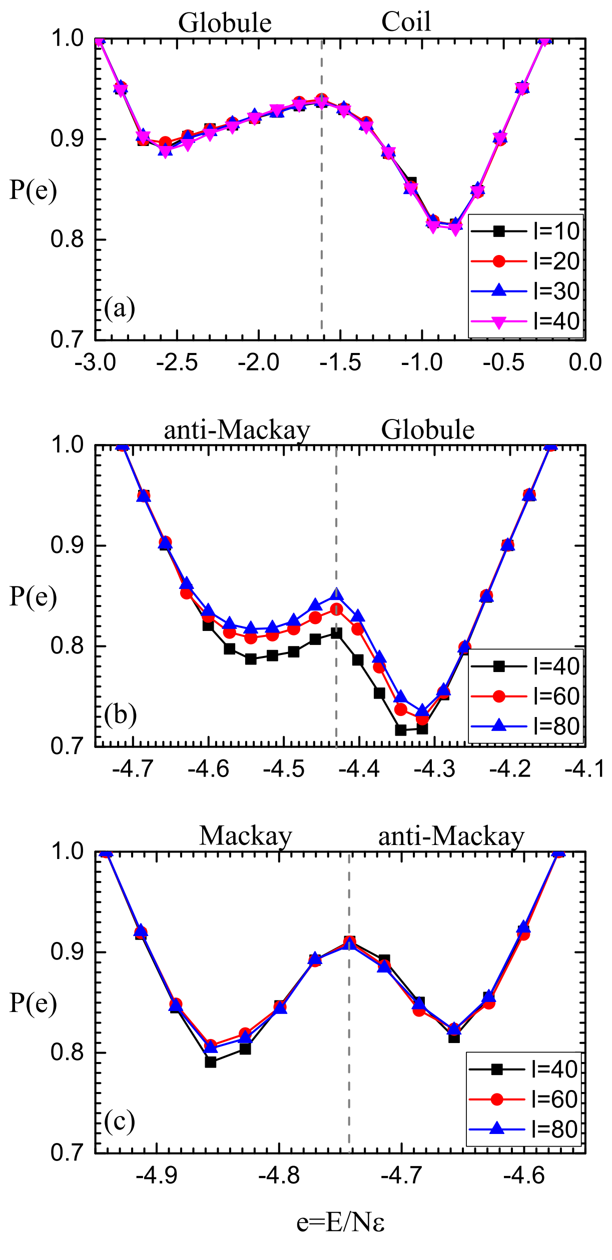

Figure 4.6: The structural transition learned by the W-shape approach in terms of input configurations pre-processed by PCA with truncation of dimensionality l as labelled in plots. The critical energy e_c positioned by the dashed lines for (a) coil-to-globule (b) globule-to-anti-Mackay and (c) anti-Mackay-to-Mackay transitions reads -1.61, -4.43 and -4.74, respectively.

configurations are sampled by Wang-Landau methods in the range of energy of interests $[e_1, e_2]$, within which a phase transition from A state to B state takes place at e_c. For the energy bins that are very far from the transition point, most of the configurations sampled belong to the same state. On the contrary, the configurations generated near the transition point could be a mixture of both A and B states. In order to locate the transition point e_c for a transition, in the W-shape scheme [170], one needs to arbitrarily specify an e. The FNN is trained with the data composed of two parts: one part is the configurations sampled within the range $[e_1, e)$, and they are labelled as A; another part is the configurations sampled within the range $(e, e_2]$, and they are labelled as B. The configurations with energy e are not used in training. Then, one can locate e_c according to the testing performance of the FNN. As one limiting case, i.e. $e = e_1$, corresponding to the situation that all data are labelled as B, the network will predict all data 100% the same with the label, so does the case $e = e_2$. For the desired case $e \approx e_c$, most of the data are labelled correctly, the network can obtain almost perfect prediction as well, due to the correct labelling. However, the performance of the network for other e positively depends on the percentage of data that are labelled correctly. That is, when e is proposed other than e_1, e_c, e_2, part of the states are mislabelled. The performance of the FNN will decrease. As a result, the performance of the FNN shows a W-shape profile.

Similar to previous works [170, 182], the FNN used in this study is composed of three layers, i.e. one input layer, one hidden layer and one output layer. Without loss of generality, the sigmoid function is adopted to activate the weighted input. Two nodes in the output layer are used to identify polymer configurations for a two-phase transition. The cross entropy is adopted as the cost function. As to the input layer, the number of nodes is manifestly determined by the dimensionality of input polymer configurations. For the raw polymer configurations, each of which consists of 102 monomers, one has to use 306 nodes in the input layer, due to three degrees of freedom for each monomer. Nevertheless, for configurations preprocessed by the PCA, the number of nodes required by the input layer is equal to l, the dimensionality of the truncated matrix. It is worth to pointing out that the representation of the input data plays an extremely crucial role in the machine learning. The more suitable data representation is adopted, the more effective a neural network can map the input data to the output data [170]. Conceptually, the number of hidden nodes strongly depends on the structural complexity of polymer configurations. In the present study, we find that 40 nodes in the hidden layer are enough to extract meaningful structural information for all four configurations.

The prediction based on the W-shape scheme to the transition points for coil-to-globule, globule-to-anti-Mackay, anti-Mackay-to-Mackay transitions, in terms of data representation in form of raw configurations and the ones pre-processed by PCA, is list in Table 4.1. Inter-

Table 4.1: A comparison of the critical energy e_c of structural transitions for coil (C), globule (G), anti-Mackay (aM) and Mackay (M) states are determined by our work and Ref. [182]. In the present work, we predict e_c by taking two types of data representation as input data, i.e. raw configurations and configurations pre-processed by PCA, respectively.

	$e_c^{\mathrm{C-G}}$	$e_c^{\mathrm{G-aM}}$	$e_c^{\mathrm{aM-M}}$
Raw configurations	-1.61 $\pm$ 0.05	-4.43 $\pm$ 0.03	-4.74 $\pm$ 0.03
PCA configurations	-1.61 $\pm$ 0.05	-4.43 $\pm$ 0.03	-4.74 $\pm$ 0.03
Work in Ref. [182]	-1.75 $\pm$ 0.05	-4.40 $\pm$ 0.03	-4.74 $\pm$ 0.03

estingly, these two different types of data representation result in the exact same prediction of the critical transition energy. It indicates that structural features of polymer configurations are well preserved by means of PCA. Further, as shown in Fig. 4.6, one observes that the effective dimensionality l in PCA, in order to precisely locate the critical transition, can approximately be truncated according to the magnitude of eigenvalues associated with the principal components. It implies that the W-shape performance benefits from the configurations conducted by PCA in advance, particularly to the polymer system with an extremely great number of monomers. In addition, as listed in Table 4.1, our results satisfactorily agree with the ones previously predicted by the supervised learning [182]. A small discrepancy exhibited on the prediction of the coil-to-globule transition is mainly ascribed to the fluctuation in the data set [182], which may suppress the sharp signal to detect the phase transition.

In addition, Xu [189] also tried to use the configurations produced by the diffusion map as the input data for the W-shape scheme. Nevertheless, the W-shape scheme fails to locate the transition points accurately. A plausible explanation would be that the delicate characteristics of polymer configurations have to be abandoned in diffusion map, for the sake of the extraction of salient features in the diffusion map method strongly rely on the definition of similarity between different states.

4.6 Summary

In this work, we utilize the unsupervised learning techniques to explore the structural transition of polymer configurations. Here, a single-chain homopolymer is studied. With the decrease of the energy, our system undergoes coil-to-globule, globule-to-anti-Mackay and anti-Mackay-to-Mackay transitions in sequence. A dimensionality reduction technique, PCA, is adopted to distinguish these four states. PCA distinguishes the coil state from the other three states, mainly ascribe to the robust extraction of the feature R_g. However, for the collapsed states with the similar size, PCA alone is not powerful enough to distinguish their difference.

We also studied a unified approach that combines the advantage of both unsupervised learning methods and neural networks. This work demonstrates that this approach has a remarkable capability of identifying the structures and structural transitions of polymeric systems.

This approach adopted here provides a generic scheme to explore the configuration transition by extracting features directly from polymer configurations themselves, without the need for any prior knowledge of order parameters, the configuration labelling in advance and complicated calculations to the thermodynamic quantities. It is desirable to extend our present scheme to the other polymeric systems undergoing complex structural transitions in the near future, such as self assembly of block copolymers, polyelectrolytes and polymer liquid crystals.

Chapter 5

Machine learning solver for the modified diffusion equation of AB diblock copolymer

5.1 Introduction

Previous chapters reveal that incorporation of machine-learning techniques [75, 137, 13, 120, 55] into computational physics to tackle physical problems have dramatically changed the classical approaches in physics. Supervised and unsupervised learning methods, with their unsurpassed capability for practical applications such as image and voice recognition, have found themselves a new playground in physics. Recent work has used machine-learning techniques to classify, manipulate, or even create the big data produced for the structural and dynamic information of various modelled systems [19, 195, 170, 182, 175, 71, 107, 166, 18, 157, 22, 23, 31, 152].

In this chapter, we explore the way of studying diblock copolymer structures by neural networks under the SCFT scheme. Note that our computational concept is very different from the traditional algorithms. We start from building a machine learning based PDE solver. For a traditional PDE solver, the functions to be determined are usually represented in some numerical form, by direct discretization or series-expansion on spectral bases; a traditional PDE solver then adjusts these numerical values to satisfy the PDEs. Here, using an FNN, we adopt a different philosophy. The functions to be solved are analytically expressed as known functions of their variables (through the connectors between the neurons) which contain undetermined parameters (such as weights and biases). During the calculation (i.e., training session), the cost functions are minimized with respect to these parameters. The final result is a universal representation of the calculated functions, but with specific parameters determined through optimization. In a sense, an FNN does not learn from the existing solutions of PDEs; they do, on the other hand, learn how to adjust themselves to satisfy the formal expression of PDEs. When searching for equilibrium structures of diblock copolymer melt, the machine learning PDE solver generates the solution of the modified diffusion equation. To solve the self-consistent equations, We need to use another FNN to represent the external fields. The training process adjusts the parameters of two FNNs simultaneously. Similarly, the minimization of the cost function with respect to the parameters of two FNNs ensures the functions generated by both FNNs satisfy self-consistent equations. Then, we can assess the density based on the propagators generated from machine learning based PDE solver, and thereby determine the structure.

5.2 Main procedure

Assume that we are dealing with well-specified, coupled differential equations for functions $q_1(\mathbf{r})$, $q_2(\mathbf{r})$, ... where the vector $\mathbf{r}$ generally represents multi-dimensional variables and could be a combination of, for example, space and time variables. Generally, we write PDEs as

$$\hat{D}_1[q_1(\mathbf{r}), q_2(\mathbf{r})...] = 0, \hat{D}_2[q_1(\mathbf{r}), q_2(\mathbf{r})...] = 0, ... \tag{5.1}$$

The differential operators, $\hat{D}_1$ and $\hat{D}_2$, act on the functions. The partial differential equations are augmented by typical "boundary conditions" (or initial conditions if time variables are involved). For example, at boundaries "1", "2", etc.,

$$\hat{B}_1[q_1(\mathbf{r}), q_2(\mathbf{r})...] = 0, \hat{B}_2[q_1(\mathbf{r}), q_2(\mathbf{r})...] = 0, ... \tag{5.2}$$

In addition, there could be constraints that govern these quantities, which are represented by

$$\hat{C}_1[q_1(\mathbf{r}), q_2(\mathbf{r})...] = 0, \hat{C}_2[q_1(\mathbf{r}), q_2(\mathbf{r})...] = 0, ... \tag{5.3}$$

These conditions could involve (usually at a lower order) further derivatives. For abbreviation, the left hand sides are denoted as $\hat{D}_1(\mathbf{r})$, $\hat{D}_2(\mathbf{r})$, $\hat{B}_1(\mathbf{r})$, $\hat{B}_2(\mathbf{r})$, $\hat{C}_1(\mathbf{r})$, $\hat{C}_2(\mathbf{r})$, etc.

In Fig. 5.1 we schematically illustrate example FNNs to be used in this chapter. At the initial stage, the parameters used in FNN are specified randomly or according to previous experience, hence, in general, the functions $q_1(\mathbf{r})$, $q_2(\mathbf{r})$, ..., calculated from the FNNs are

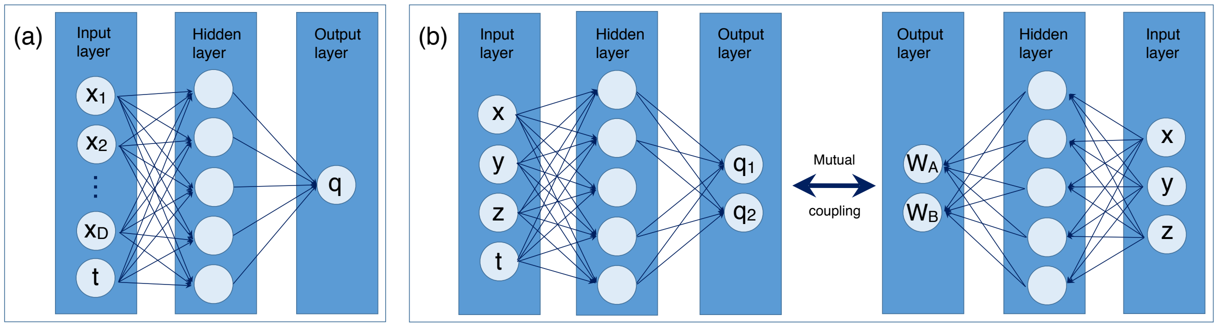

Figure 5.1: Two examples of physical problems solved here: (a) a simple diffusion equation where $q(\mathbf{x};t)$ is the density of the diffusing material in an external field at location $\mathbf{x}$ and time t and (b) complicated, coupled modified diffusion equations where $q_1(x,y,z;t)$ and $q_2(x,y,z;t)$ are the complementary reduced Green's functions for a real AB-diblock copolymer self-assembly problem, which couple to the self-consistent fields $W_A(x,y,z)$ and $W_B(x,y,z)$. In both examples, the functions to be found are represented by feed-forward neutral networks. The circles represent neuron nodes, where the input layer consists of nodes that have variables as input and the output layer are simply the functions to be determined. The connections between the input and hidden layers are assumed to be sigmoid functions and the connections between the hidden and output layers are assumed to be linear with adjustable coefficients.

far from the desirable solutions. We then design a cost function as

$$\begin{aligned} J = {} & \frac{\alpha_1}{2}\left\langle \left|\hat{D}_1(\mathbf{r})\right|^2 \right\rangle + \frac{\alpha_2}{2}\left\langle \left|\hat{D}_2(\mathbf{r})\right|^2 \right\rangle + ... \\ & + \frac{\beta_1}{2}\left\langle \left|\hat{B}_1(\mathbf{r})\right|^2 \right\rangle + \frac{\beta_2}{2}\left\langle \left|\hat{B}_2(\mathbf{r})\right|^2 \right\rangle + ... \\ & + \frac{\gamma_1}{2}\left\langle \left|\hat{C}_1(\mathbf{r})\right|^2 \right\rangle + \frac{\gamma_2}{2}\left\langle \left|\hat{C}_2(\mathbf{r})\right|^2 \right\rangle + ... \end{aligned} \tag{5.4}$$

where $\langle\cdots\rangle$ is the algebraic average of the quantity within, sampled at a set of randomly selected points in the domain of $\mathbf{r}$. Upon the minimization of J as a function of FNN parameters to reach $J = 0$, the search finds an approximation of the represented functions q_1, q_2, ... The coefficients, α_1, α_2, β_1, β_2, γ_3, γ_4, ... are penalty coefficients that can be fixed or adjusted during the minimization process. Ideally, if a minimal $J = 0$ is found, all equations in (5.1), (5.2), and (5.3) are exactly satisfied.

In a typical machine learning application, such as pattern recognition, the first input layer of an FNN is used for reading greyscale pixels of a known picture (one sample of the training data set); through FNN, the output data gives a calculated guess of the properties of the picture. During the training session, the cost function is used to minimize the difference between the guesses produced by FNN on the training data set and the known values of the desired properties. One iteration of minimization is called an epoch and (usually) a different training data set is used for the next epoch. Once the cost function converges to a value smaller than a tolerance level, the network is considered trained by supervision [13]. The progress in formulating back-propagation for the minimization process is key to computational efficiency [141].

A similar procedure is adopted here to solve differential equations. This procedure is essentially based on the stochastic gradient descent method. It consists of multiple epochs. At the beginning of an epoch, a set of coordinates **r** (with values randomly selected from its domain of interest), instead of the greyscale pixel values, are used in the input layer as a single "sample". Through FNN, the output data produces a guess of the functions to be studied. Within an epoch, many such randomly selected samples are used to produce guesses of the functions at different points in the domain. During the training session, the cost function is used to minimize the mean-square averages of the left-hand sides of Eqs. (5.1), (5.2) and (5.3), which are calculated according to the outputting, machine-guessed functions. One epoch of minimization is then performed. Once the cost function converges to zero after multiple epochs, the PDEs are considered solved, or, the FNN is trained to represent the functions that follow the PDEs, unsupervised. No numerical solutions obtained from any other methods are used in this procedure.

Note, The formulae involved in the construction and training of the solver are documented in section 2.4. In addition, appendix B provides a more concrete pedagogical guide.

5.3 Machine learning PDE solver

5.3.1 Diffusion equation

To demonstrate the machine learning PDE solver is able to find the solution, we firstly consider a simple, one-dimensional linear diffusion equation for a single density function of the diffusing material as a function of time, $q(x;t)$, in an external field $W(x)$. In reduced units, the differential equation takes the form

$$\hat{D}q(x;t) = \left[\frac{\partial}{\partial t} - \frac{1}{6}\frac{\partial^2}{\partial x^2} + W(x)\right] q(x;t) = 0. \tag{5.5}$$

To demonstrate the procedure, we assume a harmonic-potential well, $W(x) = x^2/2$, for which an exact solution exists and can be used for benchmarking. The "boundary condition" (actually an initial condition here) is simply assumed to be

$$\hat{B}_1 q(x;t) = q(x;0) - 1 = 0. \tag{5.6}$$

This particular initial condition represents a uniformly dispersed material in space at $t = 0$. The physics to be described is to switch the external field $W(x)$ on at $t = 0$, and to monitor the evolution of how the material diffuses to the central region in x [see illustration in Fig. 5.2(a)]. No further boundary conditions need to be specified for the problem.

An FNN with a single hidden layer containing $N_h = 10^2$ nodes is used. The machine-learning procedure consists of consecutive epochs. A single epoch starts from the current value of FNN parameters. Assume that we are interested in the variable domain $x = [-5, 5]$ and $t = [0, 2]$. $S = 500$ samples, i.e, 500 pairs of (x, t), are randomly selected. Among these coordinate pairs, 20% samples are selected from the "boundary" $t = 0$. The left-hand sides of Eqs. (5.5) and (5.6) are evaluated for these samples at their given values. One interesting technical note is that the derivatives needed for the evaluations all have analytic forms, which means there is no need, for example, to approximate $\partial q/\partial t$ by a finite difference, or to approximate $\partial^2 q/\partial x^2$ at the boundaries of x domain by boundary conditions. The deepest-descent direction in the FNN parameter space is determined based on the above calculation. The back-propagation algorithm [141] is utilized to adjust FNN parameters, along that direction. During this step, we set $\alpha = 1$ and $\beta_1 = 50$. We then start a new epoch based on the adjusted FNN parameters. For the current example, the tolerance level for the acceptable J value is set at $\tau = 10^{-3}$.

The machine learning solution of this toy problem is displayed in Fig. 5.2 in two views. The physical process is described by Fig. 5.2(a). At $t = 0$, the external field is switched on and the one-dimensional material begins to diffuse to the central region as t increases. The symbols in plot (b) are used to illustrate the function, selectively at a few values of x for given t. As a benchmark for comparison, the solid black curves represent the exact solution to the problem. The above example contains natural boundary conditions at $x = \pm\infty$ where $q(x;t) \to 0$. A traditional finite-difference method would require to cover the entire x domain in order to address the $\partial^2 q/\partial x^2$ term properly. Here we have selected a x domain $[-5, 5]$ to solve the PDE where the $\partial^2 q/\partial x^2$ term is analytically represented by FNN at the $x = \pm 5$. No special attention is paid to these boundaries.

In contrast, to solve the PDE in Eqs (5.5) and (5.6) in, for example, a periodic potential with period L, we need to specify an additional periodic boundary condition,

$$\hat{B}_2 q(x;t) = q(0;t) - q(L;t) = 0. \tag{5.7}$$

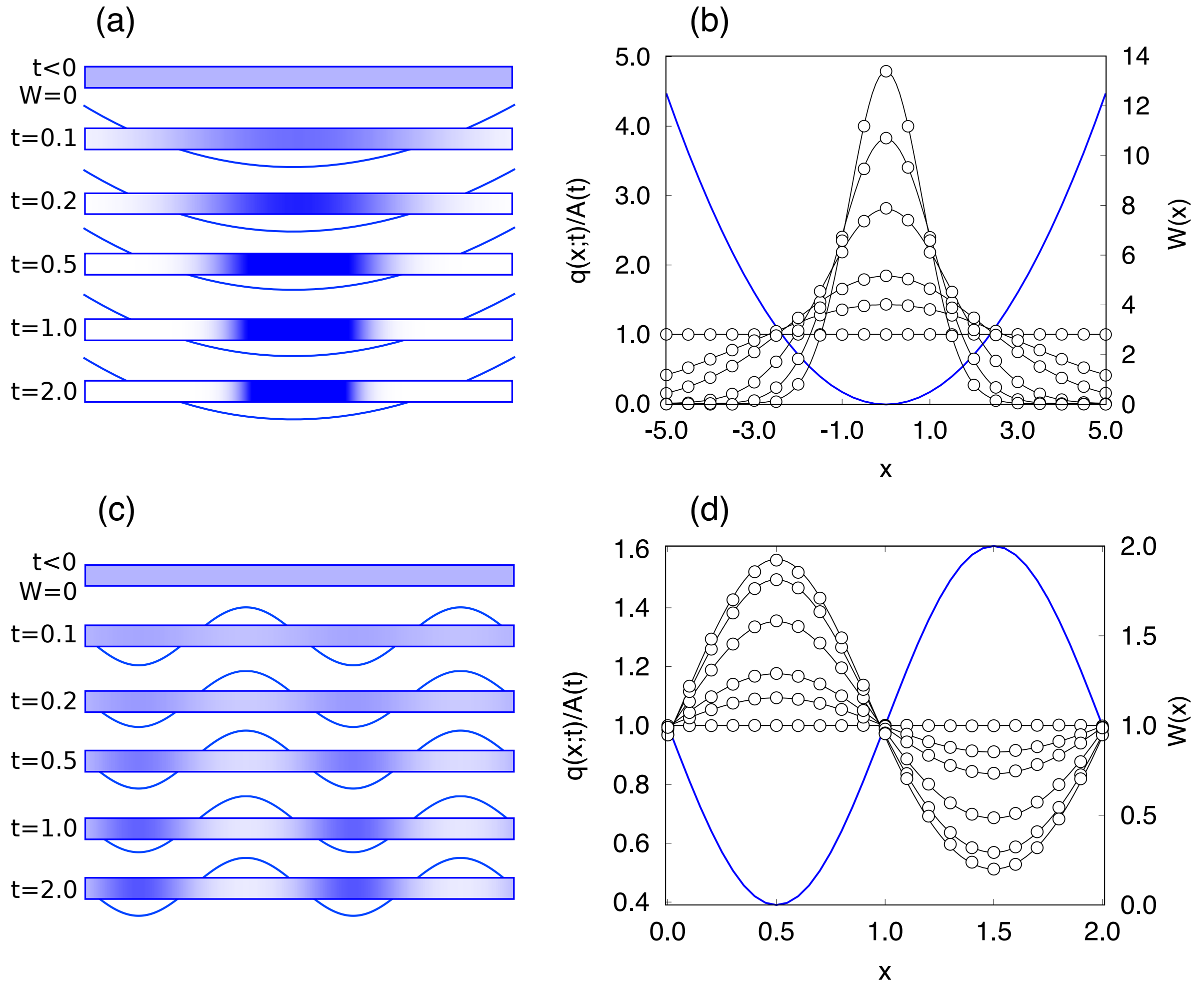

Figure 5.2: (a) Density plots of a diffusing material in a harmonic-potential well over selective times, (b) comparison between the analytical solution (solid lines) and the FNN solution (circle) at a few values of x for given $t = 0.0, 0.1, 0.2, 0.5, 1.0$, and 2.0, (c) density plots of a diffusing material in periodic-potential well over selective times, and (d) comparison between the Crank-Nicolson numerical solution (solid lines) and the FNN solution (circle) at a few values of x for given $t = 0.0, 0.1, 0.2, 0.5, 1.0, 2.0$. $A(t)$ is a normalization constant for q evaluated at each time step. The blue lines in the background, to the right scaled, are the harmonic-potential well [(a) and (b)], and the period potential well [(c) and (d)], applied on the diffusing material.

Our next toy example is letting $W(x) = \sin(\pi x)$ and we focus on the domain $x = [0, 2]$. The main computational structure remains exactly the same and the only addition is the $\hat{B}_2$ term to the cost function with a coefficient $\beta_2 = 50$. We selected $S = 600$ samples for each epoch and placed 1/6 samples to deal with $\hat{B}_1$ and another 1/6 samples $\hat{B}_2$. The solution is displayed in Figs. 5.2(c) and (d) where the symbols in (d) represent selective values of x to generate the data. The black curves in (d) are numerical solutions based on a traditional, Crank-Nicolson method [30] to solve the above PDE. A good agreement is seen between the solutions from the two methods.

It is worthwhile to mention that the final solutions found by the machine learning PDE solver will agree well with each other when training starts from different initial weights and bias. This robustness to initialization indicates we can regress the FNN used by machine learning PDE solver on an appropriate guess of the solution, so that a set of weights and bias is obtained, and then, we can use this set of weights and bias as the initial parameters of the machine learning PDE solver. This trick almost certainly will shorten the training time.

5.3.2 Performance

Generally speaking, the time it takes the machine learning PDE solver to find the solution is highly determined by the number of nodes, N_h, in the hidden layer and the error tolerance. To achieve a better performance, i.e. obtain a certain level of accuracy using less training time, we need to explore the optimal choice of N_h and set a reasonable error tolerance.

Optimal choice of the number of hidden neurons

The performance of the solver should depend both on the number of hidden nodes (the width of an FNN) and the number of hidden layers (the depth of an FNN). While we assumed one hidden layer for simplicity, N_h can affect the rate of convergence, which is related to another parameter: the tolerance level τ of the approximation by the solver. The number of minimization epochs to reach convergence depends on N_h and a pre-specified τ.

To demonstrate this point, in Fig. 5.3 we solved a $D = 1$ diffusion equation by setting three different values for τ and observe the number of epochs that the solver takes to converge. When N_h is too small, FNN is less capable, which means the solver takes a longer time to search for the solution. When N_h is too high, there are too many FNN parameters to be optimized and some may never be needed; this slows down the training. An optimal number of N_h is seen in the plot. Of course, for different mathematical problems, the

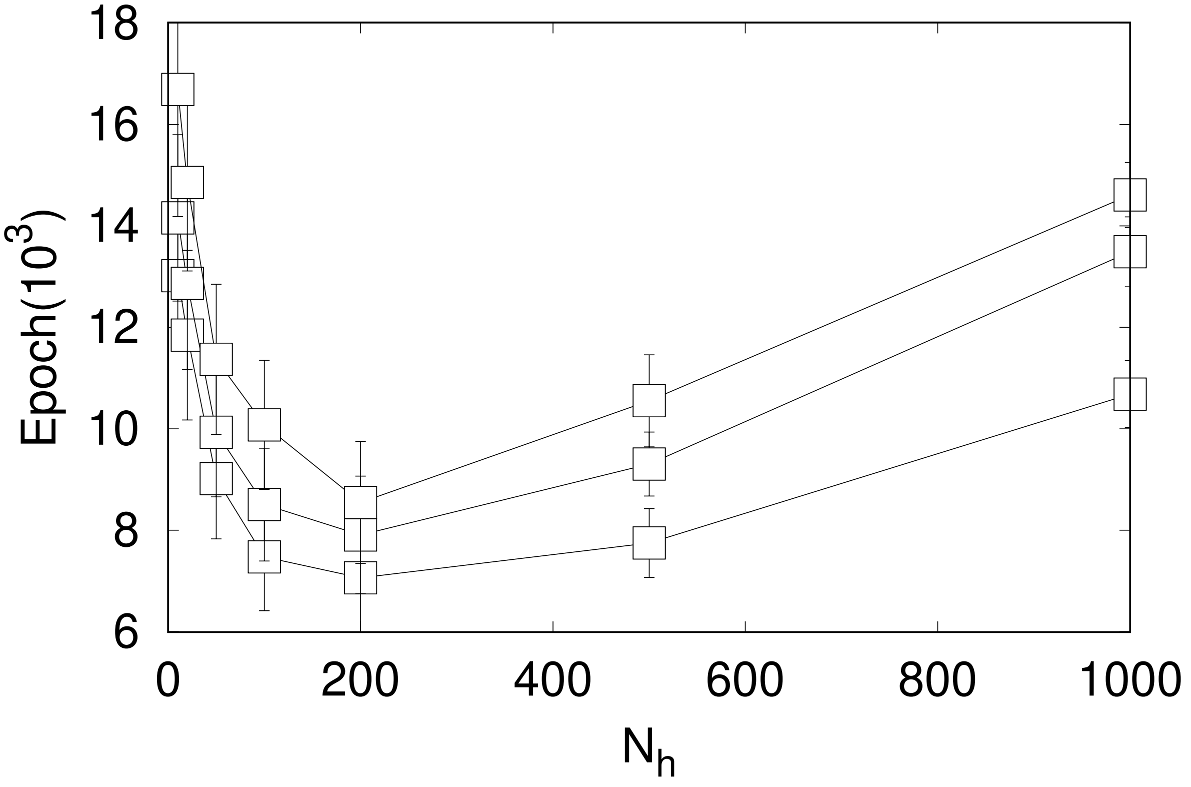

Figure 5.3: The number of epochs that the universal solver takes to reach a pre-specified error tolerance level, from bottom to top, $\tau = 10^{-2}$, 10^{-3}, and 10^{-4} respectively, as a function of the number of hidden nodes, N_h. Each data points is calculated based on 10 independent run with random parameter initialization. The one-dimensional ($D = 1$) diffusion equation is studied here.

optimal N_h may vary and usually there is a lack of *a priori* knowledge of the optimal N_h for a new problem.

Optimal choice of error tolerance

In all cases, our the requirement of error tolerance is set as 10^{-3}, and it is actually quite good. In Fig. 5.2(b) we show solution of a 1+1 dimensional solution with harmonic potential (blue curve, Fig. 5.2(b)). The circles are the reproduced FNN solutions against the exact solution represented by the solid curve. One can show that

$$q(x;t) = \sum_{n=0}^{\infty} c_n q_n(x) \mathrm{e}^{-(1/2+n)t/\sqrt{3}} \tag{5.8}$$

where $q_n(x)$, $(n = 0, 1, 2...)$ is a set of orthonormal functions,

$$q_n(x) = \left(\frac{3^{1/4}}{2^n n! \sqrt{\pi}} \right)^{1/2} H_n(3^{1/4}x) \mathrm{e}^{-\sqrt{3}x^2/2} \tag{5.9}$$

with H_n the nth-order physicist's Hermit polynomial [1]. The coefficient c_n is given by $c_n = \int_{-\infty}^{\infty} \mathrm{d}x q_n(x)$. Solid curves in Fig. 5.2 are produced by letting $t = 0.0, 0.1, 0.2, 0.5, 1.0, 2.0$ in Eq. (5.8). Visually, there is no difference in requiring $\tau = 10^{-3}$ or 10^{-5}.

5.3.3 Breaking the curse of dimensionality

One of the greatest challenges in computational complexity theory is to break the curse of dimensionality for computationally solving a problem that contains multiple variables [11]. Taking the diffusion equation for illustration, in D spatial dimensions we write

$$\hat{D}q(\mathbf{x};t) = \left[\frac{\partial}{\partial t} - \frac{1}{6} \sum_{n=1}^{D} \frac{\partial^2}{\partial x_n^2} + W(\mathbf{x}) \right] q(\mathbf{x};t) = 0. \tag{5.10}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_D)$ is a D-dimensional vector. Any traditional numerical method to solve this requires the computation to determine at least N^D representative data points. For example, the finite difference method directly divides the D-dimensional space into representative nodes, where on average N nodes for each x_n are needed. Taking an underestimate that a tradition algorithm is linear in N^D to achieve a solution of precision τ, in high-D this amounts to exponential growth in computational time and storage resource

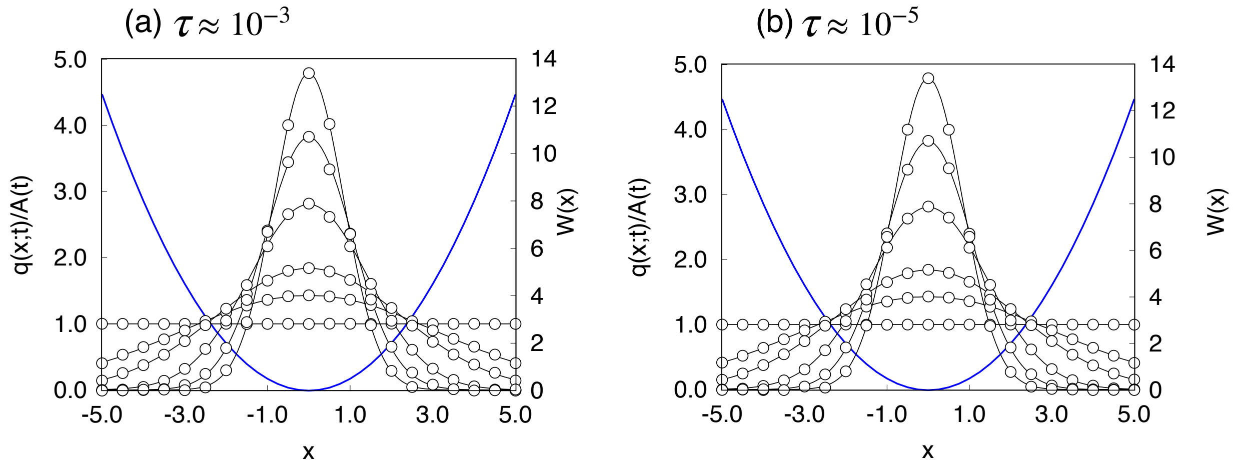

Figure 5.4: Comparison between the analytical solution (solid lines) and the FNN solution (circle) at a few values of x for given $t = 0.0, 0.1, 0.2, 0.5, 1.0$, and 2.0, when error tolerant is set at (a) 10^{-3} and (b) 10^{-5}, for a $D + 1 = 2$ problem.

[126]. In another example, the spectral-function approach requires a series-expansion of the functions in terms of well-defined spectral functions carrying D integer indices, where on average N coefficients need to be determined for each type of indices. In short, most real algorithms [132], of course, are more expensive than N^D. This problem is known as the curse of dimensionality [11].

Our universal solver takes the approach of representing functions by FNNs and turns the differential-equation solving problem into a machine-learning problem. In such a form, the number of nodes in the hidden layer, N_h, and the maximum epoch loops, M, required in a learning process to achieve a pre-specified precision τ, directly determine the computational complexity. To understand the dependence of M on D, as an example, we numerically solve Eq. (5.10) in a specific potential field $W(\mathbf{x}) = (1/2)\sum_{n=1}^{D} x_n^2$, incorporating an initial condition $\hat{B}_1 q(\mathbf{x}; 0) = q(\mathbf{x}; 0) - 1 = 0$ generalized from (5.6), for selected D up to 10^2. Other technical parameters include: for every training epoch the selection of $S = 500$ sample points [each $(D + 1)$-dimensional] in the $D + 1$ dimensional space spanned by $(x_1, x_2, \ldots, x_D; t)$, pre-specified error tolerance $\tau = 10^{-3}$ for J, and the placement of 20% of the sampling points at $t = 0$ to deal with the initial condition. To collect adequate statistics, for a given D we conducted 10 separate learning runs, each starting from a random selection of the FNN parameters. A data point in Fig. 5.5 is an averaged result from these 10 runs.

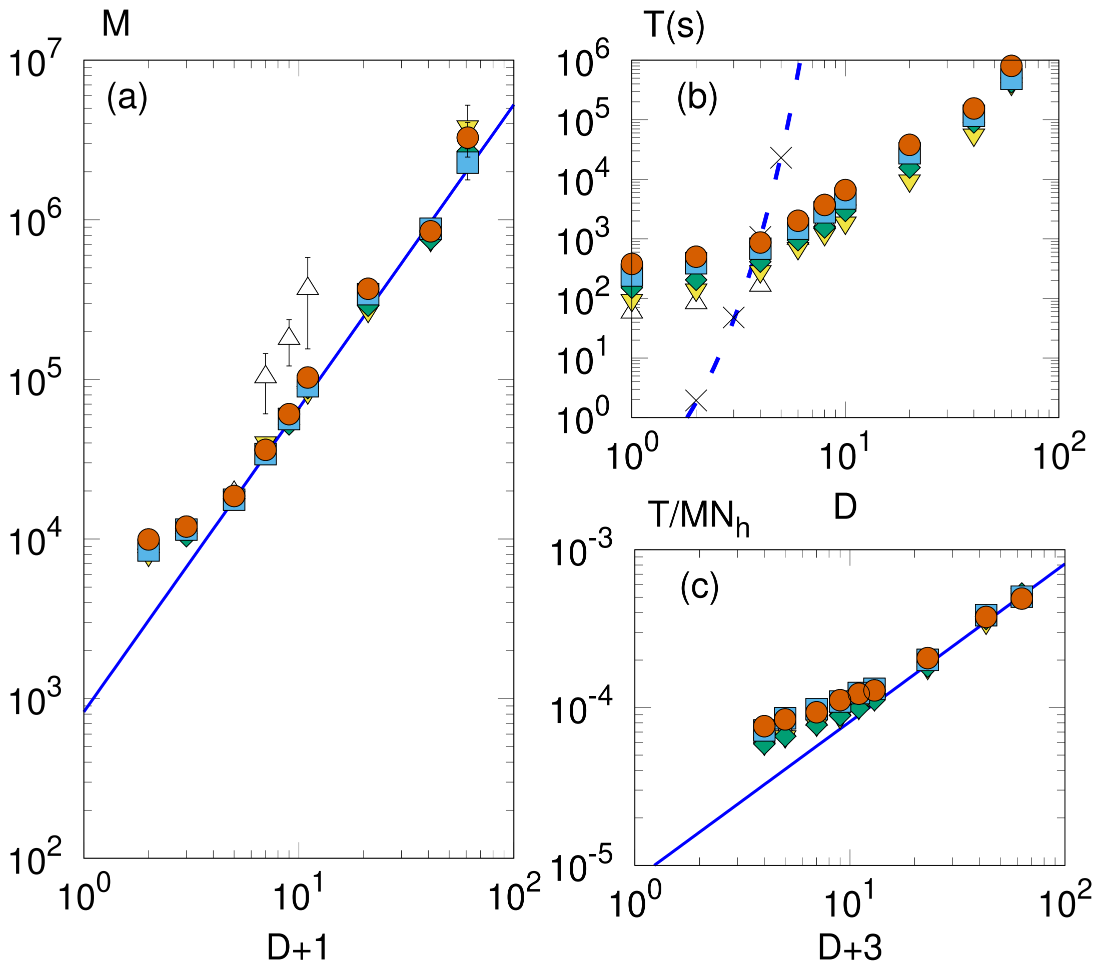

Figure 5.5: Log-log plots of (a) the maximum epochs (M), and (b) the total computational time (T) that the universal solver takes to reach the error tolerance level $\tau = 10^{-3}$, as functions of D, the number of spatial variables in a high-dimensional diffusion equation, Eq. (5.10); (c) Log-log plot of the ratio T/MN_h as a function of $D+3$. The error bars, estimated from 10 independent runs, are smaller than the plotted symbols, except for those explicitly shown. Up triangle, down triangle, diamonds, squares, and circles represent the results produced from FNNs that contain $N_h = 100, 200, 300, 400$, and 500 hidden nodes, respectively. The solid blue lines indicate the asymptotic power laws on which the data points collapse. The dashed blue curve illustrates an arbitrarily exponential dependence of the computational time expected from a traditional solver.

To explore the complexity of the problem, we used different numbers of nodes on the hidden layer, N_h. Except for the low-N_h ($= 100$) data, a striking feature of Fig. 5.5(a) is that the maximum epochs for convergence, M, follows a linear behaviour at large D on a double-logarithmic plot, with a slope $\nu \approx 1.9$,

$$M \propto D^{\nu}. \tag{5.11}$$

Although we are unable to analytically deduce this dependence, the numerical evidence indicates a rather optimistic scaling property for the required computational loops with a common exponent ν, asymptotically for large D and N_h.

Based on this relationship, we can estimate the computational resource required to solve a problem. The FNN structure is described in the section 2.4. The total number of FNN parameters $P = (D + 1) \times N_h + N_h + N_h \times 1$, where the three terms are for the number of w-parameters between the input and hidden layers, the number of b-parameters on the hidden layer, and the number of v-parameters between the hidden and output layers, respectively. An immediate advantage machine learning based PDE solve have is computational storage. Our solver memorizes $P = (D + 3)N_h$ parameters instead of approximately N^{D+1} representative nodes.

Another main concern is the computational time. On each epoch pass, P parameters need to be updated. The computational time of the back-propagation method [13] linearly depends on P, hence the total computational time T is asymptotically proportional to

$$T \propto MP = M(D + 3)N_h. \tag{5.12}$$

This is a surprisingly pleasant power-law scaling produced by the universal machine-learning solver, in comparison with the exponential law N^D illustrated in plot-(b) anticipated from a traditional approach.

5.4 Self-assembly of diblock copolymers

Having built the machine learning PDE solver, our next goal is to predict the crystallographic structures that self-assemble mesoscopically from placing many identical linear polymer chains in a finite volume [8], through using our algorithm to solve a rather complicated integrodifferential equation set for a classical computational problem in polymer physics. The SCFT is a useful tool for structural prediction of a densely packed system known as diblock copolymer melt. The system of interests contains n same diblock copolymer chains, occupying a cubic that has volume V. Each copolymer, as shown in Fig.

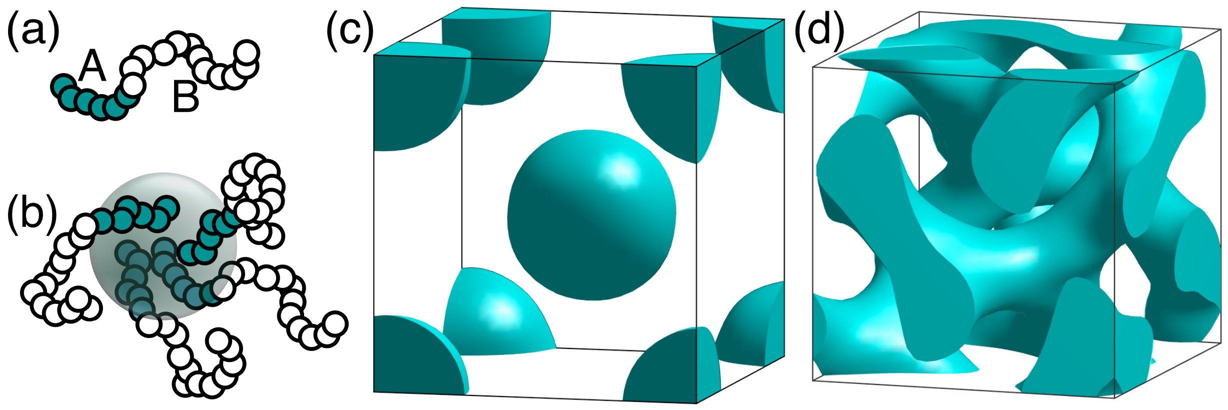

Figure 5.6: Solving the self-consistent field theory (SCFT) for the microphase structures of diblock copolymers. Plot (a) illustrates a single polymer chain where a covalent bond links A and B blocks together. Plot (b) shows the cross section of an A-rich spherical domain in a B-rich background. Plots (c) and (d) are our three-dimensional numerical solutions of the monomer-fraction profiles from the SCFT, which have body-centered cubic and gyroid structures, respectively. The solutions are obtained from a machine-learning algorithm that incorporates representations of functions conceptually shown in Fig. 5.1(b). For illustration purpose, we plot all A-rich regions with the same green color.

5.6(a), contains two blocks, consisting of A- and B-type molecular units ("monomers"), respectively represented by green and white circles. Along a polymer chain that has a total of N monomers (N is usually large), the volume ratio between A- and B-monomers is f. A pair of A- and B-monomers have a weak "dislike" energy, written in terms of the dimensionless Flory-Huggins parameter χ [72, 45]. Within the SCFT for a uniformly packed system, there are only two system parameters, f and χN.

5.4.1 Self-consistent equations for predicting the structures of diblock copolymers

The SCFT for the microphase-separated structures of a "polymer melt" consisting of many diblock copolymers is a well-established topic area in polymer physics. A significant number of researchers have contributed to this active area of research. In this thesis, we introduced the historical efforts in section 1.4. Here we list the nonlinear integrodifferential equation set that is required to solve in order to find a specific microphase-separated structure, in a form that can be used in our machine learning based solver. As a technical note, in all mathematical expressions below, the spatial variables have been scaled by the root-mean-square end-to-end distance of a polymer.

SCFT scheme:

The theory contains two physical parameters, the system-averaged, overall monomer fraction between species A and B, f, and the reduced Flory-Huggins parameter χN that measure the degree of incompatibility of the two species. The theory couples seven functions together: the monomer fraction profiles $\phi_A(\mathbf{x})$ and $\phi_B(\mathbf{x})$ for the spatial distributions of type-A and type-B monomers, the effective mean fields $W_A(\mathbf{x})$ and $W_B(\mathbf{x})$ acting on type-A and type-B monomers, the "propagators" for a single polymer chain $q_1(\mathbf{x};t)$ and its complementary $q_2(\mathbf{x};t)$, and finally, a Lagrangian multiplier $\xi(\mathbf{x})$ to enforce the a uniform density of a melt condition. The time-like variable t is an arc-variable along a linear polymer chain, comparable to the real time variable in the Feynman's path-integral formalism for quantum physics. Under SCFT scheme, $q_1(\mathbf{x};t)$ and $q_2(\mathbf{x};t)$ obey modified diffusion equations:

$$\frac{\partial q_1}{\partial t} = \left[\frac{1}{6}\nabla^2 - W(\mathbf{x};t)\right] q_1,\ q(\mathbf{x};0) = 1 \tag{5.13}$$

$$-\frac{\partial q_2}{\partial t} = \left[\frac{1}{6}\nabla^2 - W(\mathbf{x};t)\right] q_2,\ q_2(\mathbf{x};1) = 1 \tag{5.14}$$

where $W(\mathbf{x};t) = W_A(\mathbf{x})$ when $0 \le t \le f$ and $W(\mathbf{x};t) = W_B(\mathbf{x})$ when $f \le t \le 1$. $W_A(\mathbf{x})$, $W_B(\mathbf{x})$, $\phi_A(\mathbf{x})$, $\phi_B(\mathbf{x})$ and $\xi(\mathbf{x})$ obey self-consistent field equations:

$$\phi_A(\mathbf{x}) = \frac{1}{Q}\int_0^f \mathrm{d}t q_1(\mathbf{x};t) q_2(\mathbf{x};t) \tag{5.15}$$

$$\phi_B(\mathbf{x}) = \frac{1}{Q}\int_f^1 \mathrm{d}t q_1(\mathbf{x};t) q_2(\mathbf{x};t) \tag{5.16}$$

$$W_A(\mathbf{x}) = \chi N \phi_B(\mathbf{x}) + \xi(\mathbf{x}) \tag{5.17}$$

$$W_B(\mathbf{x}) = \chi N \phi_A(\mathbf{x}) + \xi(\mathbf{x}) \tag{5.18}$$

$$\phi_A(\mathbf{x}) + \phi_B(\mathbf{x}) = 1 \tag{5.19}$$

where $Q = (1/V)\int \mathrm{d}\mathbf{x} q(\mathbf{x};1)$. Under SCFT scheme, Eq. (5.13) - Eq. (5.19) can be solved self-consistently, typical algorithms are well-documented in a textbook [48].

The problem is further complicated by the fact that the free energy must be minimized by adjusting the volume. Assuming periodicity $L = V^{1/3}$, once the a self-consistent solution is obtained, we can adjust L by

$$L_{new} = L_{old} - \eta \frac{nN}{3VQ} \int \mathrm{d}\mathbf{x} \int \mathrm{d}t q_1(\mathbf{x};t) \nabla_{\mathbf{x}}^2 q_2(\mathbf{x};t), \tag{5.20}$$

where L_{new} is the new periodicity, L_{old} is the current periodicity and η is a small multiplier that adjusts how fast we change the periodicity [3].

Once a self-consistent solution is obtained, the free energy of this system can be calculated by

$$\begin{aligned}\frac{\beta}{n}F = &-\ln Q + \frac{1}{V}\int \mathrm{d}\mathbf{x}[\chi N \phi_A(\mathbf{x})\phi_B(\mathbf{x}) \\ &- W_A(\mathbf{x})\phi_A(\mathbf{x}) - W_B(\mathbf{x})\phi_B(\mathbf{x}) - \xi(\mathbf{x})(1-\phi_A(\mathbf{x}) - \phi_B(\mathbf{x}))].\end{aligned} \tag{5.21}$$

which can be used to compare the stability of different structures once the self-consistent equations are solved.

Machine learning based approach

In our machine learning based approach, we represent four of the seven functions, $W_A(\mathbf{x})$, $W_B(\mathbf{x})$, $q_1(\mathbf{x};t)$, and $q_2(\mathbf{x};t)$ by FNNs. Both $W_A(\mathbf{x})$ and $W_B(\mathbf{x})$ are functions of (x,y,z) and share a single FNN. Both $q_1(\mathbf{x};t)$ and $q_2(\mathbf{x};t)$ are functions of $(x,y,z;t)$ and share another FNN. The concept is illustrated in Fig. 5.1. They are coupled together in the differential equations and by constraints and are also connected through the definition of the other three functions which are expressed by integral relationships between these.

There are a number of constraints that are implemented in the cost function J as conceptually laid out in subsection 2.4 and section 5.2. Letting $\mathbf{r} = (\mathbf{x};t)$ we write

$$\begin{aligned}\hat{D}_1 q_1(\mathbf{r}) &= \left[\frac{\partial}{\partial t} - \frac{1}{6}\nabla^2 + W(\mathbf{x};t)\right] q_1(\mathbf{x};t) \\ \hat{D}_2 q_2(\mathbf{r}) &= \left[-\frac{\partial}{\partial t} - \frac{1}{6}\nabla^2 + W(\mathbf{x};t)\right] q_2(\mathbf{x};t)\end{aligned} \tag{5.22}$$

The cost function $J_D = \alpha \left\langle [\hat{D}_1(\mathbf{r})]^2 \right\rangle /2 + \alpha \left\langle [\hat{D}_2(\mathbf{r})]^2 \right\rangle /2$ deals with the modified diffusion equations satisfied by q_1 and q_2 by requiring the two square terms to vanish. The "initial conditions" that need to be imposed on these functions are reflected in writing

$$\begin{aligned}\hat{B}_1 q_1(\mathbf{r}) &= q_1(\mathbf{x}; t=0) - 1, \\ \hat{B}_2 q_2(\mathbf{r}) &= q_2(\mathbf{x}; t=1) - 1,\end{aligned} \tag{5.23}$$

and then incorporated in the cost function $J_{\text{ic}} = \alpha_1 \left\langle [\hat{B}_1(\mathbf{r})]^2 \right\rangle /2 + \alpha_2 \left\langle [\hat{B}_2(\mathbf{r})]^2 \right\rangle /2$. For a structure that has a periodicity L in the x-direction, the boundary conditions for $q_1(\mathbf{x};t)$ and $q_2(\mathbf{x};t)$ are considered by taking the differences

$$\begin{aligned}
\hat{B}_3 q_1(\mathbf{r}) &= q_1(0,y,z;t) - q_1(L,y,z;t),\\
\hat{B}_4 q_2(\mathbf{r}) &= q_2(0,y,z;t) - q_2(L,y,z;t),\\
\hat{B}_5 q_1(\mathbf{r}) &= ||\nabla q_1(0,y,z;t) - \nabla q_1(L,y,z;t)||,\\
\hat{B}_6 q_2(\mathbf{r}) &= ||\nabla q_2(0,y,z;t) - \nabla q_2(L,y,z;t)||.
\end{aligned} \tag{5.24}$$

where $||\cdot||$ is the module of a vector. The cost function $J_{\text{bc}} = \alpha_3 \left\langle [\hat{B}_3(\mathbf{r})]^2 \right\rangle /2 + \alpha_4 \left\langle [\hat{B}_4(\mathbf{r})]^2 \right\rangle /2 + \alpha_5 \left\langle [\hat{B}_5(\mathbf{r})]^2 \right\rangle /2 + \alpha_6 \left\langle [\hat{B}_6(\mathbf{r})]^2 \right\rangle /2$ effectively reproduces the periodic conditions when it is minimized to 0. Similar treatments are introduced for the y- and z-direction boundary conditions.

SCFT couples q_1, q_2, W_A, and W_B through relations expressed by the constraints $\hat{C}_1 = 0$ and $\hat{C}_2 = 0$, where

$$\hat{C}_1 = W_A(\mathbf{x}) - \chi N \phi_B(\mathbf{x}) - \xi(\mathbf{x}), \tag{5.25}$$

$$\hat{C}_2 = W_B(\mathbf{x}) - \chi N \phi_A(\mathbf{x}) - \xi(\mathbf{x}). \tag{5.26}$$

The functions $\phi_A(\mathbf{r})$ and $\phi_B(\mathbf{r})$ are defined by

$$\phi_A(\mathbf{x}) = \frac{1}{Q}\int_0^f \mathrm{d}t q_1(\mathbf{x};t) q_2(\mathbf{x};t), \tag{5.27}$$

$$\phi_B(\mathbf{x}) = \frac{1}{Q}\int_f^1 \mathrm{d}t q_1(\mathbf{x};t) q_2(\mathbf{x};t), \tag{5.28}$$

where the constant

$$Q = \frac{1}{V}\int \mathrm{d}\mathbf{r} q_1(\mathbf{x},1) \tag{5.29}$$

is integrated from the q functions over the considered space of volume V. The function $\xi(\mathbf{x})$, originates from the Lagrangian multiplier of the incompressibility condition, is connected to the above quantities by

$$\xi(\mathbf{x}) = [W_A(\mathbf{x}) + W_B(\mathbf{x}) - \chi N]/2. \tag{5.30}$$

The constraints $\hat{C}_1 = 0$ and $\hat{C}_2 = 0$ are effectively implemented in our machine-learning solver through minimizing the cost function $J_C = \gamma[\hat{C}_1]^2/2 + \gamma[\hat{C}_1]^2/2$.

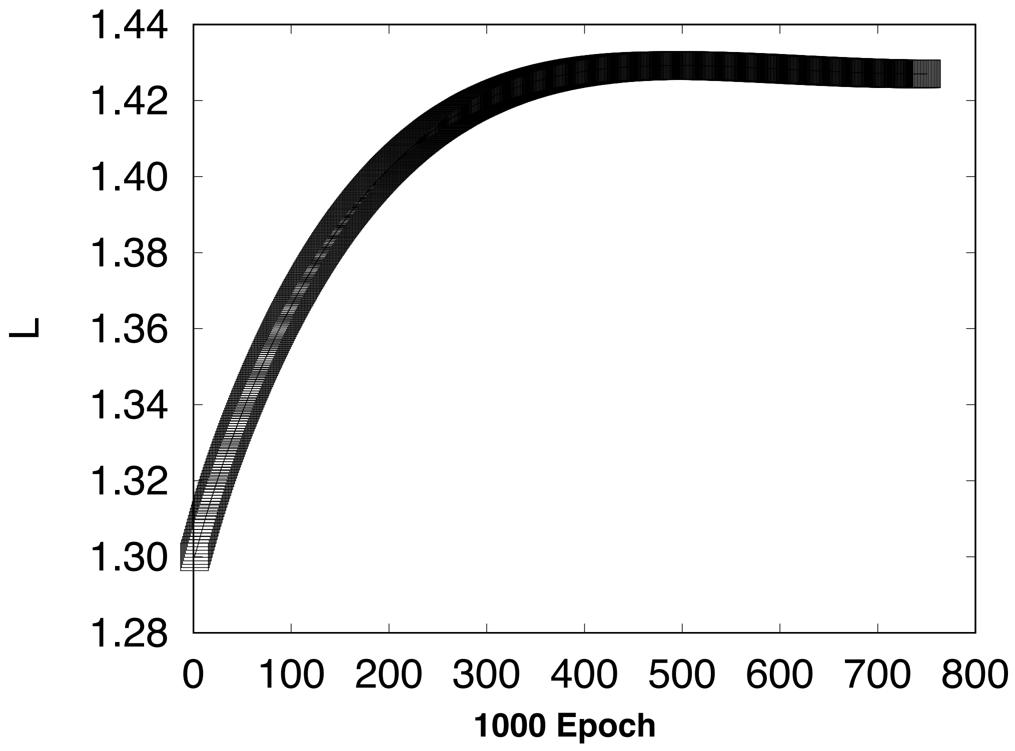

Figure 5.7: Plot of L with respect to epoch.

The total cost function J is the sum

$$J = J_D + J_{\text{ic}} + J_{\text{bc}} + J_C \tag{5.31}$$

and is minimized in the learning process, with $\alpha = 1$, $\beta_1 = \beta_2 = 1000$, $\beta_3 = \beta_4 = 200$, and $\gamma = 200$. The minimization procedure consists of consecutive epochs. In one single epoch, $S = 7 \times 10^3$ points are randomly selected in the variable domain and fed into both FNNs to generate all required function values and derivatives used in the cost function. The coefficients of the penalty terms were selected by trial and error; the relative magnitudes decide the prioritization of minimizing the related terms. Normally, to emphasize the auxiliary conditions, we use coefficients that are two orders of magnitude higher than the coefficients for the partial differential equations themselves.

We adjust L every 1000 epochs following Eq. (5.20). As an example, the behaviour of L with respect to epoch when $f = 0.5$ and $\chi N = 12$ is plotted in Fig. 5.7 When L stables, a structure is considered as found.

5.4.2 Equilibrium structure observed

The chemically different A- and B-type monomers drives the diblock copolymer system to phase-separate into A- and B-rich spatial domains, but only the micro-phase separation happens due to the covalent bond that connects A and B blocks links the two phase-separated domains together within the dimension of a typical polymer size. Figure 5.6(b) illustrates how A- and B-rich spatial domains can form in a multi-chain system. The delicate balance of the interaction energy and entropic stretching energy of the A and B blocks results in the formation of various geometric domains with size in the nanometer range [5, 115, 98, 99, 84, 83]. Two well-known 3D structures i.e. body-centered cubic (bcc) sphere phase ($Q_{Im\bar{3}m}$) and complex bicontinuous cubic (gyroid) phase ($Q_{Ia\bar{3}d}$), as an example, calculated from theories and observed in a wealth of works are shown in Fig. 5.6(c) and Fig. 5.6(d), within a unit cell.

Basically, four basic, unknown functions $q_1(\mathbf{r})$, $q_2(\mathbf{r})$, $W_A(\mathbf{x})$, and $W_B(\mathbf{x})$, must be found numerically for a given $[f, \chi N]$ pair. In a traditional approach, multiple iterations are needed to achieve the self-consistency of the solution set. The main idea is to propose a guess for the external fields $W_A(\mathbf{x})$ and $W_B(\mathbf{x})$, which are used in the diffusion-like equations governing the propagator functions $q_1(\mathbf{r})$ and $q_2(\mathbf{r})$. Then, integrating over the t variable step by step, one obtains the solutions for $q_1(\mathbf{r})$ and $q_2(\mathbf{r})$. The external fields $W_A(\mathbf{x})$ and $W_B(\mathbf{x})$ are then updated according to these solutions and a new iteration step starts. Self-consistency is obtained after multiple loops of iterations, at which point the W fields converge.

A completely different philosophy is adopted here. In order to implement our new machine-learning solver, we propose that the four functions are presented by two FNNs, conceptually shown in Fig. 5.1(b). The learning is done by looping through epochs. At every epoch, the FNNs learn the new profiles of these four functions simultaneously by updating the FNN parameters, according to the minimization requirement of the cost function. The cost function itself contains terms that are targeted at solving the diffusion-like equations, that effectively deal with the boundary conditions for given t and given $\mathbf{x}$, and that couple W_A, W_B with q_1, q_2 nonlinearly in an integral form. There is no need to integrate the differential equations over t-range step by step, because t is now treated at an equal footing as $\mathbf{x}$. Most importantly, the iteration loop that updates $W_A(\mathbf{x})$ and $W_B(\mathbf{x})$ step-by-step is now eliminated. The self-consistency is directly enforced through the non-linear coupling of the four functions. The heart of this theoretical approach is to solve SCFT equations in order to yield structural prediction.

Starting from a random choice of the FNN parameters, the FNNs finally converge to one of the metastable or stable structures allowed for a given $[f, \chi N]$ set. We reproduced all known structures, such as those presented in Ref. [49, 115] and selectively present two in Fig. 5.6(c). These three-dimensional bcc structure [5.6(c)] and gyroid structure [Fig. 5.6(d)] can be obtained from the machine-learning solver at $[f, \chi N] = [0.28, 16]$ and $[0.375, 16]$, respectively. There is a good agreement between our and previous solutions.

5.5 Discussion

In this chapter, we demonstrated a new way of solving equilibrium states of diblock copolymer melt based on machine learning based PDE solver. As of now, the free energy converges to the second decimal point, and a typical run needs hours, which is orders of magnitude slower than the traditional method. The accuracy and efficiency of this solver are needed to be improved in the future. Two promising improvements are using deep neural networks to generate the unknown functions and employing more advanced training methods, such as second-order methods [9]. We hope the solver can be more accurate and faster in the future.

5.6 Summary

Taking advantage of the universal approximation theorem, in this work we present a machine-learning procedure designed to solve PDEs typically seen in theoretical physics.

We started by introducing a fundamental diffusion problem. The mathematical problem of finding a solution now becomes finding the FNN parameters with optimization. The computational time required to solve a problem now depends on iterations to optimize network parameters. We demonstrated, through a particular example, how the computational time scales as the number of variables in a problem and the number of hidden nodes. One interesting finding is that the number of epochs required to achieve convergence follows a power law with a universal exponent *independent* of the number of hidden nodes. A direct consequence is that the computational time of a problem with a large number of variables can now be efficiently handled by the universal solver. As a function of the number of variables, the anticipated exponentially large computational time in a traditional method is now replaced by a faster power law. Thus, we expect that this universal solver is particularly useful for physical problems containing many variables.

Then, we employed an adapted version of machine learning PDE solver to solve for diblock copolymer states by tackling a complicated integrodifferential equation set produced from the SCFT. We observed a good agreement between our solution and the traditional solution. Searching for stable and meta-stable conformations of a polymeric system is a long enduring and crucial topic in soft matter physic in large part due to their rich and complex self-assembly behaviour [7, 185, 112, 116, 191, 178]. The heart of a theoretical approach is to solve SCFT equations in order to make a structural prediction. Here we wish the machine-learning solver can overcome the main hurdle encountered in a conventional method – the stability of a proposed algorithm.

Chapter 6

Conclusion and Outlook

6.1 Conclusion

In this thesis, we explored some machine learning techniques that can be employed to study structural properties of polymeric systems.

In chapter 1 and chapter 2, the large number of historical efforts on studying polymer structures are reviewed. The recent inspirations we get from machine learning and machine learning techniques we utilized are summarized.

In chapter 3, we studied the capability of a supervised learning method on identifying polymer states and transition points. We demonstrated that a well trained FNN can identify structures, such as coil, globule, anti-Mackay and Mackay conveniently, merely depends on the monomer coordinates, even when the difference between structures is minuscule. This ability of structural identification can be transfer to configurations sampled from different models without re-train the neural network. This method can also be adapted to identify transition points accurately. One advantage of this approach is directly sending the configurational data represented by molecular coordinates to a neural network, without defining order parameters or calculating the heat capacity, which are conventionally used in computer simulations to rigorously determine a transition point.

In chapter 4, we explored the capability of unsupervised learning methods on identifying polymer states and transitions points. Two methods, PCA and a unified approach are studied. We found that PCA can distinguish random coil from collapsed states through dimensional reduction and visualization, mainly due to the feature R_g is extracted from configurations. Although PCA is widely considered as the simplest unsupervised learning technique and preserves only linear information in projection, the results presented persuasively reveals its effectiveness. Moreover, we also find that PCA is not sophisticated enough to distinguish the difference between globule, anti-Mackay and Mackay states, which suggest that in order to classify collapse states with similar R_g, we need to consult more powerful methods. On the other hand, the unified approach shows a strong ability in accurately locating the transition points in an unsupervised learning manner. We also find the prediction of the transition points could highly benefit from the data representation. The low dimensional representations obtained from PCA facilitate the training of the neural network, particularly to the polymeric system with high polymerization, mainly ascribe to many redundant details are eliminated through the pre-treatment of data according to linear dimensionality reduction.

These methods introduced in chapter 3 and chapter 4 are generic and have the potential to be utilized in future studies. In the study of phase transitions, machine learning methods provide a way to identify structures without designing an order parameter, and a way to locate the transition point without calculating heat capacity in the first place. In particular, in the low-energy (or low-temperature) regime, a simulated system often encounters potential-energy traps which need to be treated by using a non-Boltzmann weight. Taking the Wang-Landau algorithm as an example, the calculation of the heat capacity in the extremely low-energy regime requires high numerical precision of the computed density of states, which is achievable but requires extensive computations. The neural network process describes here, on the other hand, does not require such precision, as long as independent configurations used for supervised training are produced by numerical simulation methods.

In chapter 5, we developed a machine learning PDE solver and demonstrated how to find equilibrium structures of diblock copolymer through solving integrodifferential equations in SCFT. For our machine learning PDE solver, a striking feature we found is that the computational cost follows a power-law instead of exponential, which means this solver is potentially helpful for physical problems involving high-dimensional PDEs. Our solver avoids the potential pitfalls typically seen in a traditional approach to solve PDEs. The approximations for the derivative operators in the PDEs are no longer needs and all required information is expressed by analytic expressions, through the representing FNNs. The approximation made in a traditional method highly influences the stability of a typical algorithm in such a way that the stability of a computational algorithm usually becomes the main concern. Here, this difficulty is avoided by turning the solution-finder problem into a machine learning problem. The solver is an unsupervised procedure that requires no prior information of the solution and accommodates boundary conditions and constraints systematically. In terms of searching for new structures, our solver provides a new way to solve self-consistent equations, and to generate equilibrium structures of diblock copolymers given f and χN. The structures obtained from our approach have good agreements with traditional methods.

6.2 Outlook

The machine learning methods we studied open plenty of opportunities for future works. The purpose of this section is to outlines some possible future studies.

6.2.1 Identifying polymer states

- *Exploring the usage of more advanced neural networks:*
  In Ref. [182], we only explored the capability of FNN. There are more advanced neural network architectures, such as long short-term memory (LSTM), residual networks, are believed to be more powerful than FNN [55]. Their potential for identifying polymer states haven't yet been fully explored. Recently, the LSTM has been incorporated in our approach and applied to classify confined off-lattice rod-like molecules. We successfully identified multiple types of topological defects induced by frustrations [171]. This success encourages us to further explore the capability of different kinds of neural networks.

- *Utilize a better representation of input data:*
  As pointed out in chapter 4, the representation of the input data is very important. Taking Ref. [54] as an example. Geiger et al. investigated detecting local polymorphism in Lennard-Jones fluid. In this work, raw coordinates of particles are encoded into a set of symmetry functions before fed into an FNN. These symmetry functions are designed to be rotationally, translationally and permutationally invariant. Using symmetry functions as input, one does not have to generate training samples oriented along various directions, nor need to shift, re-scale samples. In the meantime, this conversion reduces the dimensionality of input data to the number of symmetry functions used. Nevertheless, finding a proper representation, albeit feasible in this work, is a nontrivial task in general. for polymeric systems, a promising tool is bond orientational order (BOO) parameters [160, 193]. Finding a proper representation for machine learning methods via BOO is rather unstudied. Using BOO as input may decrease the number of training samples needed and lead to a more efficient training.

- *Identifying order of a transition:*
  We find it is hard for machine learning methods to determine the order of a transition. Hopefully, this problem could be overcome by future studies.

6.2.2 Machine learning based PDE solver

- *Application in worm-like chain model:*
  Given the fact that machine learning PDE solver is more competitive for high dimensional problems, we expect our approach can be applied to more complex systems. For example, in order to find equilibrium structures of worm-like diblock copolymers, the modified diffusion equation we need to solve, under the SCFT scheme, is a six dimensional diffusion like equation. Simply solving this equation under a given chemical field by traditional methods is time and computer memory consuming [77], let alone solving it self-consistently. As a next step, we hope our approach can be developed and applied to the worm-like chain models as a tool to study the phase behaviours [20, 77, 37]. We believe our approach will reduce the computational cost on solving self-consistent equations of the worm-like chain models, and providing plenty of opportunities in searching for more equilibrium structures. In addition, we hope that complications involving stability analysis of a finite-difference method, for instance in solving a PDE or self-consistent equations, no longer the concern.

- *PDEs of other types:*
  By far, machine learning based PDE solver is mostly applied to solve parabolic PDEs, such as Schrodinger's equations [135], modified diffusion equation [181], or Black-Scholes PDEs [156]. In comparison, Karniadakis et al. [135] demonstrated that machine learning based PDE solver is able to solve Burger's equation, which is one kind of hyperbolic PDE. Nevertheless, The study on the extensibility of this approach to other hyperbolic PDEs is limited and worth spending efforts in the future.

- *etc.Stochastic differential equation:*
  Solving stochastic differential equations (SDE) is crucial to many studies in polymer physics, one of which is using molecular dynamics to study polymer states [12]. The research in this direction is at a preliminary stage. We hope this area can get more attention.

In summary, the studies presented in this thesis reflect that machine learning methods provide exciting new ways to study polymer morphology, such as using the unified approach to identify transition points and simultaneously discover order parameters without prior knowledge of the structures themselves, and using neural network to approximate the solution of PDEs and generating equilibrium states of diblock copolymer melts. We anticipate machine learning methods are going to play an increasingly important role in polymer physics, and in the meantime we hope our works can stimulate other ideas in future researches or real-life applications.

APPENDICES

Appendix A

Polymer models and Monte Carlo methods

A.1 Gaussian-chain model with a square-well potential

In the text, the first model we used is a polymer made of monomers that are connected by spring potentials. The reduced bonded Hamiltonian of the polymer is

$$\beta E_{\text{bond}} = \frac{3}{2a^2} \sum_{i=1}^{N-1} \left(\mathbf{r}_i - \mathbf{r}_{i+1}\right)^2, \tag{A.1}$$

where $\beta = 1/k_{\text{B}}T$ is the Boltzmann factor. The system interaction potential energy is

$$E_{\text{int}} = \frac{1}{2} \sum_{ij} U(|\mathbf{r}_{ij}|) \tag{A.2}$$

where $\mathbf{r}_{ij}$ is the distance vector between monomer-i and monomer-j. The pair-wise interaction potential has the form,

$$U(r) = \begin{cases} \infty & \text{if } 0 \le r^2 < (0.9a)^2, \\ -\epsilon & \text{if } (0.9a)^2 \le r^2 \le 2a^2, \\ 0 & \text{otherwise.} \end{cases} \tag{A.3}$$

We refer to this model as GSM in the text.

A.2 FENE model with a LJ interaction

In the text, we used the second polymer model which has a system energy E as the sum of the bonded and interaction energies. The former is described by

$$E_{\text{bond}} = \sum_{i=1}^{N-1} E_{\text{FENE}}\left(|\mathbf{r}_i - \mathbf{r}_{i+1}|\right), \tag{A.4}$$

where E_{FENE} is a particular realization of the FENE model,

$$E_{\text{FENE}}(r) = -20\epsilon R^2 \ln[1 - (r - r_0)/R)^2], \tag{A.5}$$

in which r is the distance between two connected monomers, $R = 0.3b$ controls the bond-length variations, and $r_0 = 0.7b$. The interaction potential energy formally follow (A.2) but the two body interaction is replaced by a truncated Lennard-Jones potential,

$$U(r) = \begin{cases} U_{\text{LJ}}(r) - U_{\text{LJ}}(r_{\text{c}}) & \text{if } 0 \le r < r_{\text{c}} \\ 0 & \text{otherwise,} \end{cases} \tag{A.6}$$

where $r_{\text{c}} = 2.5\sigma$. The Lennard-Jones potential has the standard form,

$$U(r) = 4\epsilon \left[\left(\frac{r_m}{r}\right)^{12} - \left(\frac{r_m}{r}\right)^{6} \right], \tag{A.7}$$

where $r_m = 2^{-1/6} r_0$ and ϵ measures the potential-well depth (which is adjusted by the value at the truncation point).

A.3 Monte Carlo methods

For a polymeric system with a Hamiltonian $H(\mathbf{r})$, all the statistical information is in partition function defined as

$$Z = \sum \exp\{-\beta H(\mathbf{r})\} \tag{A.8}$$

where the summation takes over all possible configurations, $\beta = k_B T$ is inverse temperature, k_B is Boltzmann constant, T is temperature. A thermodynamic quantity, O, usually can be calculated by

$$\langle O \rangle = \frac{1}{Z} \sum O(\mathbf{r}) \exp\{-\beta H(\mathbf{r})\} \tag{A.9}$$

where $\exp\{-\beta H(\mathbf{r})\}/Z$ is called Boltzmann factor. Nevertheless, the number of possible configurations grows exponentially with chain polymerization, which makes the direct evaluation of O usually unfeasible. On the contrary, If we view $O(\mathbf{r})$ as an observation and view $p(O(\mathbf{r})) = \exp\{-\beta H(\mathbf{r})\}/Z$ as the probability of this observation occurs. If we can obtain samples of $O(\mathbf{r})$ following a probability distribution $p(O(\mathbf{r}))$, then O can be estimated by

$$\langle O \rangle = \sum_i O_i(\mathbf{r}) p(O_i(\mathbf{r})) \tag{A.10}$$

where $O_i(\mathbf{r})$ represents the value of O calculated based on the ith sample. This is just the basic idea of Monte Carlo simulation. In short, Monte Carlo method refers broadly as a class of algorithms that obtain numerical solutions through random sampling. It was originally proposed by Fermi, Ulam, von Neumann, Metropolis [119] in the 1930s-1940s, and then developed to polymer physics by a lot of pioneers, Binder [91], Landau [174]. When utilizing Monte Carlo simulations to study coarse-grained polymer models, two versions of Monte Carlo methods, Metropolis method and Wang-Landau method, are very commonly used. In the subsequent subsections, we will describe the basic concepts of these methods, the application of Metropolis methods to sample configurations of the GSM, and the application of Wang-Landau methods to sample configurations of the FENE model.

A.3.1 Metropolis method

The direct calculations of statistical quantities are usually unachievable, partially due to there are too many configurations to sum over in partition function. However, the Monte Carlo method allows one to find the equilibrium distribution following $\exp\{-\beta H(\mathbf{r})\}$, and then the statistical quantities of interests can be estimated by the ensemble averages [50]. Taking Monte Carlo simulation of a polymer chain, with Hamiltonian $H(\mathbf{r})$, as an example. Metropolis method aims at generating random samples sequentially from a distribution $p(\mathbf{r})$, read

$$p(\mathbf{r}) \propto \exp\{-\beta H(\mathbf{r})\} \tag{A.11}$$

Here, $p(\mathbf{r})$ is determined up to a constant multiplier. It introduced a transition probability from current state $\mathbf{r}$ to a proposed trial state $\mathbf{r}'$, to evolve a random guess distribution to $p(\mathbf{r})$ after algorithm reaches stationary. Assuming that the transition probability from a state $\mathbf{r}$ to a proposed new state $\mathbf{r}'$ is $T_{\mathbf{r}\mathbf{r}'}$, then the dynamic function of $p(\mathbf{r})$ is given by

$$\frac{\partial p(\mathbf{r})}{\partial t} = \sum_{\mathbf{r}\neq\mathbf{r}'} (T_{\mathbf{r}'\mathbf{r}} p(\mathbf{r}') - T_{\mathbf{r}\mathbf{r}'} p(\mathbf{r})) \tag{A.12}$$

The equilibrium will be reached when $\partial p(\mathbf{r})/\partial t = 0$. This equilibrium condition indicates that if the polymer chain is currently at state $\mathbf{r}_t$, and Metropolis method proposes a move $\mathbf{r}_{t+1}$, then trial move will be accepted or reject based on acceptance ratio:

$$j = T_{\mathbf{r}_t \mathbf{r}_{t+1}}/T_{\mathbf{r}_{t+1}\mathbf{r}_t} = p(\mathbf{r}_{t+1})/p(\mathbf{r}_t) = \min\{1, \exp\{-\beta(H(\mathbf{r}_{t+1}) - H(\mathbf{r}_t))\}\} \tag{A.13}$$

If the proposed move is accepted, the current state is set as $\mathbf{r}_{t+1}$, otherwise, the current state remains as $\mathbf{r}_t$. We can keep proposing new states and recording the current state once for a while. Then, one can calculate the quantities of interest, O, as

$$\langle O \rangle = \frac{1}{L}\sum_{i=1}^{L} O(\mathbf{r}_i) \tag{A.14}$$

In our work, we use Metropolis method the sample configurations from GSM. The configurations and measurements used in Figs. 3.4(a), (b), (c), and Fig. 3.5 in chapter 3 were produced from Monte Carlo runs in which the Boltzmann weight $W = \exp(-\beta E_{\text{bond}} - \beta E_{\text{int}})$, where the two potential energies are expressed in Eqs. (A.1) and (A.2), were used. Every Monte Carlo step (MCS) contains N repeated Monte Carlo trial moves of locally displaced monomers. For every specified $k_{\text{B}}T/\epsilon$, 10^7 MCS was used in the initial equilibration and 5×10^3 configurations were recorded in a production run comprised of $2{\times}10^8$ MCS, taken with a lapse of $4{\times}10^4$ MCS. The specific heat, which is defined by:

$$\tilde{C} = \left(\langle E_{\text{int}}^2 \rangle - \langle E_{\text{int}} \rangle^2\right)/\epsilon^2 \tag{A.15}$$

In both cases, the Monte Carlo average $\langle \ldots \rangle$ was performed in the production run.

A.3.2 Wang-Landau method

In the above sections, we assumed all different configurations are independent of each other. However, that independency is not guaranteed. In Metropolis method, to avoid the bias introduced from highly correlated configurations, samples can only be taken longer than every other t_{ac} steps, where t_{ac} is the autocorrelation time, defined as

$$t_{ac} = \frac{\sum_{t=1}^{\infty}\left(\langle O(t_0)O(t_0+t)\rangle - \langle O \rangle^2\right)}{\langle O^2 \rangle - \langle O \rangle^2} \tag{A.16}$$

where t_0 means the current sample step. The larger t_{ac} is, the longer we need to wait between two consecutive samples. Unfortunately, t_{ac} usually divergent at the transition point of a second order transition, and this phenomenon is called critical slowing down. When accurately locate phase transition points is crucial, the sampling in near transition point is very inefficient due to the long waiting. Besides, polymer systems usually involves complicated energy landscape that has deep local minima. The acceptance rate in Eq. A.13 may be very low when random sampling trying to go across the barrier between local minima, which makes sampling inefficient as well.

To overcome these disadvantages, in 2001, Wang and Landau[174, 173] proposed an algorithm that utilize the density of states, $g(e)$, a quantity describes the number of states available in an energy level over an energy interval, to calculate thermodynamic quantities of interest. This algorithm relies on an observation that a flat histogram of energy distribution will generated if one random sample states in energy space with the probability inverse proportional to the density of states. In another word, it uses a weight factor $w(e)$ to form a flat histogram $h(e)$ at different energy levels, read

$$w(e)g(e)\exp(-\beta e) \propto h(e) = \text{constant} \tag{A.17}$$

When equilibrium, $w(e) \propto 1/g(e)$. At the beginning, Wang-Landau algorithm set $g(e)$ as 1, and histogram $H(e)$ as 0 for all e. Then it performs a series of loops. Inside one loop, when the system state is $\mathbf{r}_t$ at current time, and it is proposed to move to $\mathbf{r}_{t+1}$, one have energy $e_t = e(\mathbf{r}_t)$ and $e_{t+1} = e(\mathbf{r}_{t+1})$. The acceptance probability in Wang-Landau method is inversely proportional to $g(e)$, which gives an acceptance ratio

$$j = w(e_{t+1})/w(e_t) = \min\{1, g(e_t)/g(e_{t+1})\} \tag{A.18}$$

Once the new state is determined, $g(e)$ and $H(e)$ can be updated following:

$$\begin{aligned} g(e) &= g(e) \times c \\ H(e) &= H(e) + 1 \end{aligned} \tag{A.19}$$

c is an adjustable constant, usually set as $\exp\{1\}$ at the beginning. This loop ends when $H(e)$ is flat, and c will be updated to $\sqrt{c}$. Wang-Landau method is considered stabled when c is close enough to 1, and $g(e)$ is considered as converged to the desired density. Afterwards, the partition function can be calculated following

$$Z = \sum_e g(e)\mathrm{e}^{-\beta e} \tag{A.20}$$

and the the quantities of interest, O, can be calculated thereby. For example, when study the phase transitions of the FENE model, the quantities of interests is inverse temperature $\beta(e)$ and its derivative $\gamma(e)$. Inverse temperature usually defined as

$$\beta(e) = T^{-1}(e) = (\frac{\mathrm{d}S}{\mathrm{d}e})_{N,V} \tag{A.21}$$

where e is the system energy per monomer, $S(e) = k_B \ln g(e)$ is microcanonical entropy given energy e. The derivation of inverse temperature is given as

$$\gamma(e) = \frac{\mathrm{d}\beta(e)}{\mathrm{d}e} = \frac{\mathrm{d}^2 S}{\mathrm{d}e^2} \tag{A.22}$$

$\beta(e)$ and $\gamma(e)$ offered a systematical way to classify phase transitions. The transition points locates at the peak of $\gamma(e)$. For the first order transitions, the slop of $\beta(e)$ at the transition point is positive, or the peak value of $\gamma(e)$ is positive. For the second order transitions, the slop of $\beta(e)$ at the transition point is negative, or the peak value of $\gamma(e)$ is negative.

Using the Wang-Landau algorithm, we determined the density of states of the FENE model, which was used for the calculation of $\beta(e)$ and $\gamma(e)$ both defined and discussed in Ref. [146] in Figs. 3.4(d), 3.4 in chapter 3, and Fig. A.1 in Appendix A.2, over a wide range of energy space by conducting two series of simulations, one covering $e = [-5, -4]$ and the other $[-3, 2]$. In total 30 energy bins were used in each simulation. We used the procedure described in Refs. [174, 173], with a final modification-factor $f_{\text{final}} = \exp(2^{-29})$ to produce high-precision data. One small revision is made to the original procedure; when the inverse density of states is used as the Monte Carlo transition weight, linear interpolations are introduced to connect the logarithmic values of the density of states at the centers of adjacent energy bins; this is in contrast to the original histogram-type weight scheme.

Configurations used in these figures were produced from a production run consisting of 10^9MCS, in which the inverse density of states was used as the Monte Carlo transition weight. Approximately 5×10^3 configurations were recorded at every energy bin. The inverse temperature $\beta(e)$ and its derivative $\gamma(e)$ defined in a microcanonical ensemble, and calculated from sampled configurations are shown in Fig. A.1, produced from Monte Carlo simulations following the Wang-Landau algorithm [174, 173]. The left panel is similar to Fig. 1 in Ref. [146], but shifts in both e and vertical directions are noticeable. These shifts do not affect the physics we discuss in this paper. The current parametrization exactly follows the description in Ref. [146], except for the length scale b included here for accounting purpose; it is unknown where these shifts come from.

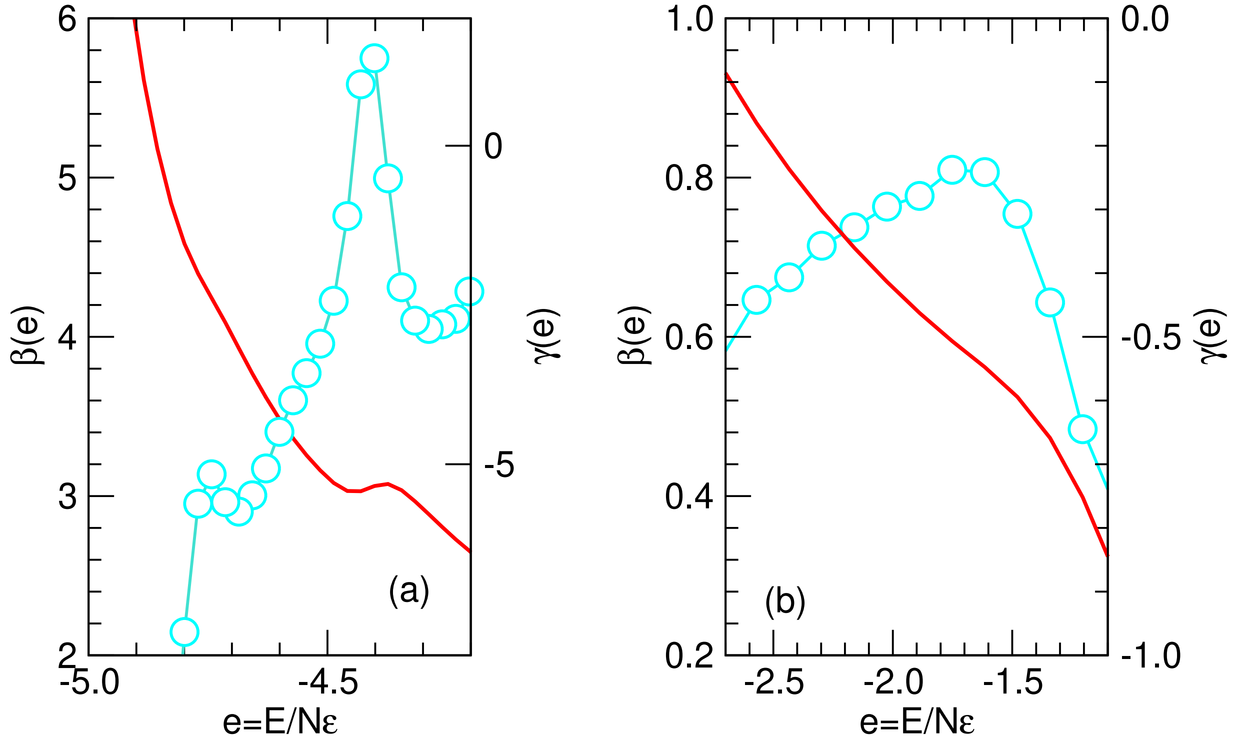

Figure A.1: Inverse temperature $\beta(e)$ (red, to the left scale) and its derivative $\gamma(e)$ (light blue, to the right scale) as functions of reduced energy per monomer $e = E/N\epsilon$, in (a) low- and (b) midlow-energy regimes. The peaks in the heat-capacity-like $\gamma(e)$ separate different polymer phases: coil, globule, anti-Mackay and Mackey (from high- to low-e regimes). The FENE model was used in Wang-Landau Monteo Carlo simulations to produce this figure.

Appendix B

A pedagogical guide of the machine learning PDE solver

To provide a detailed illustration on how to build and train a machine learning PDE solver, under a pedagogical purpose, we use a toy initial value problem:

$$\frac{\partial q}{\partial t} = c\frac{\partial}{\partial x^2}q \tag{B.1}$$

with the boundary condition and initial condition

$$q(0;t) = 0,\ q(L;t) = 0,\ q(x;0) = \psi(x) \tag{B.2}$$

as an example to specify technical details of machine learning PDE solver. In this appendix, the FNN we use is exactly the same as the one in Fig. 2.4, with two input neurons, N_h hidden neurons and one output neuron. This FNN has three sets of parameters:

- $\mathbf{w}$: weights connect input and hidden neuron, a $N_h \times 2$ matrix
- $\mathbf{b}$: biases fed into hidden neuron, a $N_h \times 1$ vector
- $\mathbf{v}$: weights connect hidden and output neuron, a $1 \times N_h$ matrix

For a single input $(x_n; t_n)$, where the first column stores the spatial coordinate, x_n, sampled from the domain of interests, the second column stores the corresponding timestamp, t_n, the output of the FNN is

$$q(x_n; t_n) = \sum_{j=1}^{N_h} v_j \sigma(z_j(x_n; t_n)), \tag{B.3}$$

where, $z_j(x_n; t_n) = w_{j1}x_n + w_{j2}t_n + b_j$, σ is the activation function

$$\sigma(z_j(x_n; t_n)) = [1 + \exp(-z_j(x_n; t_n))]^{-1}. \tag{B.4}$$

For simplicity, $\sigma(z_j(x_n; t_n))$ will be noted as σ_{jn}.

In Eq. (B.3), all partial derivatives needed in the PDE to be solved can be calculated analytically. For example

$$\frac{\partial q(x_n; t_n)}{\partial t_n} = \sum_{j=1}^{N_h} v_j \sigma_{jn}(1 - \sigma_{jn}) w_{j2} \tag{B.5}$$

Second order partial derivatives can be taken this way as well but the analytic expressions are omitted here.

Assuming that N_0 inputs are sampled within the domain of interest, N_1 inputs are sampled from the boundary $x = 0$, N_2 inputs are sampled from the boundary $x = L$ and N_3 inputs are sampled at initial time $t = 0$. In this case, the specific form of cost function is given by:

$$\begin{aligned} J =& \frac{1}{2N_0} \sum_{n=1}^{N_0} \left[\left(\frac{\partial}{\partial s} - c\frac{\partial}{\partial x^2} \right) q(x_n; t_n) \right]^2 \\ &+ \frac{1}{2N_1} \sum_{n_1=1}^{N_1} q(0; t_{n_1})^2 + \frac{1}{2N_2} \sum_{n_2=1}^{N_2} q(L; t_{n_2})^2 \\ &+ \frac{1}{2N_3} \sum_{n_3=1}^{N_3} [q(x_{n_3}; 0) - \psi(x_{n_3})]^2 \end{aligned} \tag{B.6}$$

Defining:

$$\begin{aligned} \Delta_n =& \left(\frac{\partial}{\partial s} - c\frac{\partial}{\partial x^2} \right) q(x_n; t_n) \\ =& \sum_j v_j \sigma_{jn}(1 - \sigma_{jn}) w_{j2} \\ &- \sum_j c v_j \sigma_{jn}(1 - \sigma_{jn})(1 - 2\sigma_{jn}) w_{j1}^2, \end{aligned} \tag{B.7}$$

the derivative of cost with respect to weights and biases are given by:

$$
\begin{aligned}
\frac{\partial J}{\partial w_{jk}} &= \frac{1}{N_0}\sum_{n=1}^{N_0}\Delta_n\{v_j\sigma_{jn}(1-\sigma_{jn})(1-2\sigma_{jn})r_{nk}w_{j2} + v_j\sigma_{jn}(1-\sigma_{jn})\delta_{k2} \\
&\quad - c[v_j\sigma_{jn}(1-\sigma_{jn})(1-6\sigma_{jn}+6\sigma_{jn}^2)r_{nk}w_{j1}^2 + 2v_j\sigma_{jn}(1-\sigma_{jn})(1-2\sigma_{jn})w_{j1}\delta_{k1}]\} \\
&\quad + \frac{1}{N_1}\sum_{n_1=1}^{N_1} q(0;t_{n_1})v_j\sigma_{jn_1}(1-\sigma_{jn_1})r_{n_1k} + \frac{1}{N_2}\sum_{n_2=1}^{N_2} q(L;t_{n_2})v_j\sigma_{jn_2}(1-\sigma_{jn_2})r_{n_2k} \\
&\quad + \frac{1}{N_3}\sum_{n_3=1}^{N_3}\left[q(x_{n_3};0)-\psi(x_{n_3})\right]v_j\sigma_{jn_3}(1-\sigma_{jn_3})r_{n_3k} \\
\frac{\partial J}{\partial b_j} &= \frac{1}{N_0}\sum_{n=1}^{N_0}\Delta_n\{v_j\sigma_{jn}(1-\sigma_{jn})(1-2\sigma_{jn})w_{j2} - cv_j\sigma_{jn}(1-\sigma_{jn})(1-6\sigma_{jn}+6\sigma_{jn}^2)w_{j1}^2\} \\
&\quad + \frac{1}{N_1}\sum_{n_1=1}^{N_1} q(0;t_{n_1})v_j\sigma_{jn_1}(1-\sigma_{jn_1}) + \frac{1}{N_2}\sum_{n_2=1}^{N_2} q(L;t_{n_2})v_j\sigma_{jn_2}(1-\sigma_{jn_2}) \\
&\quad + \frac{1}{N_3}\sum_{n_3=1}^{N_3}\left[q(x_{n_3};0)-\psi(x_{n_3})\right]v_j\sigma_{jn_3}(1-\sigma_{jn_3}) \\
\frac{\partial J}{\partial v_j} &= \frac{1}{N_0}\sum_{n=1}^{N_0}\Delta_n\{\sigma_{jn}(1-\sigma_{jn})w_{j2} - c\sigma_{jn}(1-\sigma_{jn})(1-2\sigma_{jn})w_{j1}^2\} \\
&\quad + \frac{1}{N_1}\sum_{n_1=1}^{N_1} q(0;t_{n_1})\sigma_{jn_1} + \frac{1}{N_2}\sum_{n_2=1}^{N_2} q(L;t_{n_2})\sigma_{jn_2} + \frac{1}{N_3}\sum_{n_3=1}^{N_3}\left[q(x_{n_3};0)-\psi(x_{n_3})\right]\sigma_{jn_3}
\end{aligned}
\tag{B.8}
$$

In Eq. (B.8), subscript j ranges from 1 to N_h, k ranges from 1 to 2. r_{n1}, r_{n2} represents x_n, t_n respectively. δ is the Kronecker delta.

At this stage, we have all the formula necessary for constructing and training a machine learning PDE solver.

Appendix C

Publications

1. Michael Walters, **Qianshi Wei**, Jeff Z. Y. Chen, *Phys. Rev. E*, **99**, 062701 (2019);
2. Xin Xu, **Qianshi Wei**, Huaping Li, Yuguo Chen, Yuzhang Wang, Ying Jiang, *Phys. Rev. E*, **99**, 043307 (2019);
3. **Qianshi Wei**, Ying Jiang, Jeff Z. Y. Chen, *Phys. Rev. E*, **98**, 053304 (2018);
4. **Qianshi Wei**, Roger G Melko, Jeff Z. Y. Chen, *Phys. Rev. E*, **95**, 032504 (2017);

References

[1] Milton Abramowitz and Irene A Stegun. *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*. Courier Corporation, 1964.

[2] Louis-François Arsenault, Alejandro Lopez-Bezanilla, O Anatole von Lilienfeld, and Andrew J Millis. Machine learning for many-body physics: The case of the anderson impurity model. *Phys. Rev. B*, 90(15):155136, 2014.

[3] Jean-Louis Barrat, Glenn H. Fredrickson, and Scott W. Sides. Introducing variable cell shape methods in field theory simulations of polymers. *The Journal of Physical Chemistry B*, 109(14):6694–6700, 2005. PMID: 16851752.

[4] Christopher M. Bates and Frank S. Bates. 50th anniversary perspective: Block polymers—pure potential. *Macromolecules*, 50(1):3–22, 2017.

[5] F. S. Bates, M. F. Schulz, A. K. Khandpur, S. Förster, J. H. Rosedale, K. Almdal, and K. Mortensen. Fluctuations, conformational asymmetry and block copolymer phase behaviour. *Faraday Discussions*, 98:7, 1994.

[6] Frank S Bates. Polymer-polymer phase behavior. *Science*, 251(4996):898–905, 1991.

[7] Frank S. Bates and Glenn H. Fredrickson. Block copolymer thermodynamics: Theory and experiment. *Annual Review of Physical Chemistry*, 41(1):525–557, 1990.

[8] Frank S. Bates, Marc A. Hillmyer, Timothy P. Lodge, Christopher M. Bates, Kris T. Delaney, and Glenn H. Fredrickson. Multiblock polymers: Panacea or pandora's box? *Science*, 336(6080):434–440, 2012.

[9] Roberto Battiti. First-and second-order methods for learning: between steepest descent and newton's method. *Neural computation*, 4(2):141–166, 1992.

[10] Jörg Behler and Michele Parrinello. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Phys. Rev. Lett.*, 98:146401, Apr 2007.

[11] Richard Bellman. *Dynamic programming.* Princeton University Press, 1957.

[12] Kurt Binder. *Monte Carlo and molecular dynamics simulations in polymer science.* Oxford University Press, 1995.

[13] C Bishop. *Pattern Recognition and Machine Learning.* Springer, New York, 2007.

[14] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. pages 144–152, 1992.

[15] Peter Broecker, Juan Carrasquilla, Roger G Melko, and Simon Trebst. Machine learning quantum phases of matter beyond the fermion sign problem. *Sci. Rep.*, 7(1):8823, 2017.

[16] Murray Campbell, A.Joseph Hoane, and Feng hsiung Hsu. Deep blue. *Artificial Intelligence*, 134(1):57 – 83, 2002.

[17] Jaime G Carbonell, Ryszard S Michalski, and Tom M Mitchell. An overview of machine learning. In *Mach Learn*, pages 3–23. Springer, 1983.

[18] Giuseppe Carleo and Matthias Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355(6325):602–606, 2017.

[19] Juan Carrasquilla and Roger G. Melko. Machine learning phases of matter. *Nature Physics*, 13:431, Feb 2017.

[20] Jeff Z.Y. Chen. Theory of wormlike polymer chains in confinement. *Progress in Polymer Science*, 54-55:3 – 46, 2016. The Effects of Confinement on Polymeric Thermal Transitions and Nanostructuring.

[21] Stephen ZD Cheng. *Phase transitions in polymers: the role of metastable states.* Elsevier, 2008.

[22] Kelvin Ch'ng, Juan Carrasquilla, Roger G. Melko, and Ehsan Khatami. Machine learning phases of strongly correlated fermions. *Phys. Rev. X*, 7:031038, Aug 2017.

[23] Kelvin Ch'ng, Nick Vazquez, and Ehsan Khatami. Unsupervised machine learning account of magnetic transitions in the hubbard model. *Phys. Rev. E*, 97:013306, Jan 2018.

[24] Ronald R. Coifman and Stéphane Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5 – 30, 2006. Special Issue: Diffusion Maps and Wavelets.

[25] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sep 1995.

[26] Natanael C Costa, Wenjian Hu, ZJ Bai, Richard T Scalettar, and Rajiv RP Singh. Principal component analysis for fermionic critical points. *Phys. Rev. B*, 96(19):195138, 2017.

[27] JP Cotton, B Farnoux, and G Jannink. Neutron diffraction in dilute and semidilute polymer solutions. *The Journal of Chemical Physics*, 57(1):290–294, 1972.

[28] Richard Courant and David Hilbert. *Methods of Mathematical Physics: Partial Differential Equations.* John Wiley & Sons, 2008.

[29] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theor.*, 13(1):21–27, September 2006.

[30] J. Crank and P. Nicolson. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Advances in Computational Mathematics*, 6(1):207–226, Dec 1996.

[31] E. D. Cubuk, S. S. Schoenholz, J. M. Rieser, B. D. Malone, J. Rottler, D. J. Durian, E. Kaxiras, and A. J. Liu. Identifying structural flow defects in disordered solids using machine-learning methods. *Phys. Rev. Lett.*, 114:108001, Mar 2015.

[32] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, Dec 1989.

[33] Christos Davatzikos, Kosha Ruparel, Yong Fan, DG Shen, M Acharyya, JW Loughead, RC Gur, and Daniel D Langleben. Classifying spatial patterns of brain activity with machine learning methods: application to lie detection. *Neuroimage*, 28(3):663–668, 2005.

[34] PG De Gennes. Collapse of a polymer chain in poor solvents. *Journal de Physique Lettres*, 36(3):55–57, 1975.

[35] Pierre-Gilles De Gennes. *Scaling concepts in polymer physics.* Cornell university press, 1979.

[36] P Debye. Molecular-weight determination by light scattering. *The Journal of Physical Chemistry*, 51(1):18–32, 1947.

[37] Mingge Deng, Ying Jiang, Haojun Liang, and Jeff Z. Y. Chen. Wormlike polymer brush: A self-consistent field treatment. *Macromolecules*, 43(7):3455–3464, 2010.

[38] Caroline Desgranges and Jerome Delhommelle. A new approach for the prediction of partition functions using machine learning techniques. *The Journal of Chemical Physics*, 149(4):044118, 2018.

[39] M. Doi and S. F. Edwards. *The Theory of Polymer Dynamics.* Oxford Univ. Press, New York, 1986.

[40] Fran çois Drolet and Glenn Fredrickson. Combinatorial screening of complex block copolymer assembly with self-consistent field theory. *Phys. Rev. Lett.*, 83:4317–4320, Nov 1999.

[41] Weinan E, Jiequn Han, and Arnulf Jentzen. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, 5(4):349–380, Dec 2017.

[42] Sam F Edwards. The statistical mechanics of polymers with excluded volume. *Proceedings of the Physical Society*, 85(4):613, 1965.

[43] Paul Ehrenfest and Tatiana Ehrenfest. *The conceptual foundations of the statistical approach in mechanics.* Courier Corporation, 2002.

[44] A Ferguson. Machine learning and data science in soft materials engineering. *J. Phys.: Condens. Matter*, 30:043002, 2018.

[45] Paul J. Flory. Thermodynamics of high polymer solutions. *The Journal of Chemical Physics*, 10(1):51–61, 1942.

[46] Paul J Flory. *Principles of polymer chemistry.* Cornell University Press, 1953.

[47] William J Frawley, Gregory Piatetsky-Shapiro, and Christopher J Matheus. Knowledge discovery in databases: An overview. *AI magazine*, 13(3):57, 1992.

[48] Glenn Fredrickson. *The equilibrium theory of inhomogeneous polymers.* Oxford University Press, 2006.

[49] Glenn H. Fredrickson, Venkat Ganesan, and François Drolet. Field-theoretic computer simulation methods for polymers and complex fluids. *Macromolecules*, 35(1):16–39, 2002.

[50] Daan Frenkel and Berend Smit. *Understanding molecular simulation: from algorithms to applications*, volume 1. Elsevier, 2001.

[51] Matteo Frigo and Steven G Johnson. The design and implementation of fftw3. *Proceedings of the IEEE*, 93(2):216–231, 2005.

[52] Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[53] Kunihiko Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural Net*, 1(2):119–130, 1988.

[54] Philipp Geiger and Christoph Dellago. Neural networks for local structure detection in polymorphic systems. *The Journal of Chemical Physics*, 139(16):164105, 2013.

[55] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, 2016.

[56] Jiayi Guo, Haojun Liang, and Zhen-Gang Wang. Coil-to-globule transition by dissipative particle dynamics simulation. *The Journal of chemical physics*, 134(24):244904, 2011.

[57] Zuojun Guo, Guojie Zhang, Feng Qiu, Hongdong Zhang, Yuliang Yang, and An-Chang Shi. Discovering ordered phases of block copolymers: New results from a generic fourier-space approach. *Phys. Rev. Lett.*, 101:028301, Jul 2008.

[58] Ian W Hamley. *Developments in block copolymer science and technology.* Wiley, New York, 2004.

[59] Jiequn Han, Arnulf Jentzen, and E Weinan. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

[60] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 04 1970.

[61] Simon Haykin. *Neural Networks: A Comprehensive Foundation (3rd Edition)*. Prentice-Hall, Inc., 2007.

[62] Eugene Helfand. Theory of inhomogeneous polymers: Fundamentals of the gaussian random-walk model. *The Journal of Chemical Physics*, 62(3):999–1005, 1975.

[63] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Process. Mag.*, 29(6):82–97, 2012.

[64] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[65] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 278–282 vol.1, Aug 1995.

[66] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, Aug 1998.

[67] K. M. Hong and J. Noolandi. Theory of inhomogeneous multicomponent polymer systems. *Macromolecules*, 14(3):727–736, 1981.

[68] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251 – 257, 1991.

[69] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359 – 366, 1989.

[70] Wenjian Hu, Rajiv RP Singh, and Richard T Scalettar. Discovering phases, phase transitions, and crossovers through unsupervised machine learning: A critical examination. *Phys. Rev. E*, 95(6):062122, 2017.

[71] Li Huang and Lei Wang. Accelerated monte carlo simulations with restricted boltzmann machines. *Phys. Rev. B*, 95:035105, Jan 2017.

[72] Maurice L. Huggins. Solutions of long chain compounds. *The Journal of Chemical Physics*, 9(5):440–440, 1941.

[73] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, 2010.

[74] Anil K Jain, Robert P. W. Duin, and Jianchang Mao. Statistical pattern recognition: A review. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(1):4–37, 2000.

[75] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning.* Springer, 2013.

[76] Arnulf Jentzen, Diyora Salimova, and Timo Welti. A proof that deep artificial neural networks overcome the curse of dimensionality in the numerical approximation of kolmogorov partial differential equations with constant diffusion and nonlinear drift coefficients. *arXiv preprint arXiv:1809.07321*, 2018.

[77] Ying Jiang and Jeff Z. Y. Chen. Influence of chain rigidity on the phase behavior of wormlike diblock copolymers. *Phys. Rev. Lett.*, 110:138305, Mar 2013.

[78] I. T Jolliffe. *Principal Component Analysis.* Wiley, Chichester, U.K., 2002.

[79] MI Jordan and TM Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[80] Josh P. Kemp and Zheng Yu Chen. Formation of helical states in wormlike polymer chains. *Phys. Rev. Lett.*, 81:3880–3883, Nov 1998.

[81] Alexei R Khokhlov. *Statistical physics of macromolecules.* Amer Inst of Physics, 1994.

[82] AR Khokhlov and AN Semenov. On the theory of liquid-crystalline ordering of polymer chains with limited flexibility. *Journal of Statistical Physics*, 38(1-2):161–182, 1985.

[83] Kyungtae Kim, Akash Arora, Ronald M. Lewis, Meijiao Liu, Weihua Li, An-Chang Shi, Kevin D. Dorfman, and Frank S. Bates. Origins of low-symmetry phases in asymmetric diblock copolymer melts. *Proceedings of the National Academy of Sciences*, 115(5):847–854, 2018.

[84] Kyungtae Kim, Morgan W. Schulze, Akash Arora, Ronald M. Lewis, Marc A. Hillmyer, Kevin D. Dorfman, and Frank S. Bates. Thermal processing of diblock copolymer melts mimics metallurgy. *Science*, 356(6337):520–523, 2017.

[85] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[86] RG Kirste, WA Kruse, and K Ibel. Determination of the conformation of polymers in the amorphous solid state and in concentrated solution by neutron diffraction. *Polymer*, 16(2):120–124, 1975.

[87] Peter Kotelenez. *Stochastic ordinary and stochastic partial differential equations: transition from microscopic to macroscopic equations*, volume 58. Springer-Verlag New York, 2008.

[88] Manoj Kumar and Neha Yadav. Multilayer perceptrons and radial basis function neural network methods for the solution of differential equations: a survey. *Computers & Mathematics with Applications*, 62(10):3796–3811, 2011.

[89] Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, 1998.

[90] Isaac E Lagaris, Aristidis C Likas, and Dimitris G Papageorgiou. Neural-network methods for boundary value problems with irregular boundaries. *IEEE Transactions on Neural Networks*, 11(5):1041–1049, 2000.

[91] David P Landau and Kurt Binder. *A guide to Monte Carlo simulations in statistical physics*. Cambridge university press, 2014.

[92] Lev Davidovich Landau. On the theory of phase transitions. *Ukr. J. Phys.*, 11:19–32, 1937.

[93] Mohamed Laradji, An-Chang Shi, Rashmi C. Desai, and Jaan Noolandi. Stability of ordered phases in weakly segregated diblock copolymer systems. *Phys. Rev. Lett.*, 78:2577–2580, Mar 1997.

[94] Y. LeCun, C. Cortes, and C Burges. The mnist database of handwritten digits. `http://web.archive.org/web/20080207010024/http://www.808multimedia.com/winnt/kernel.htm`. Accessed: 2010-09-30.

[95] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[96] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.

[97] Yann LeCun, LD Jackel, Léon Bottou, Corinna Cortes, John S Denker, Harris Drucker, Isabelle Guyon, UA Muller, E Sackinger, Patrice Simard, et al. Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective*, 261:276, 1995.

[98] Sangwoo Lee, Michael J. Bluemle, and Frank S. Bates. Discovery of a frank-kasper ŠÒ phase in sphere-forming block copolymer melts. *Science*, 330(6002):349–353, 2010.

[99] Sangwoo Lee, Chris Leighton, and Frank S. Bates. Sphericity and symmetry breaking in the formation of frank–kasper phases from one component materials. *Proceedings of the National Academy of Sciences*, 111(50):17723–17731, 2014.

[100] Ludwik Leibler. Theory of microphase separation in block copolymers. *Macromolecules*, 13(6):1602–1617, 1980.

[101] C-D Li, D-R Tan, and F-J Jiang. Applications of neural networks to the studies of phase transitions of two-dimensional potts models. *Ann. Phys.*, 391:312–331, 2018.

[102] Jianfeng Li, Hongdong Zhang, and Jeff Z. Y. Chen. Structural prediction and inverse design by a strongly correlated neural network. *Phys. Rev. Lett.*, 123, Sep 2019.

[103] Mingqi Li and Christopher K. Ober. Block copolymer patterns and templates. *Materials Today*, 9(9):30 – 39, 2006.

[104] Henry W. Lin and Max Tegmark. Why does deep and cheap learning work so well? 2016.

[105] Cheng-Yuan Liou, Jau-Chi Huang, and Wen-Chie Yang. Modeling word perception using the elman network. *Neurocomputing*, 71(16-18):3150–3157, 2008.

[106] Richard P Lippmann. Review of neural networks for speech recognition. *Neural Comput*, 1(1):1–38, 1989.

[107] Junwei Liu, Yang Qi, Zi Yang Meng, and Liang Fu. Self-learning monte carlo method. *Phys. Rev. B*, 95:041101, Jan 2017.

[108] Ye-Hua Liu and Evert P. L. van Nieuwenburg. Discriminative cooperative networks for detecting phase transitions. *Phys. Rev. Lett.*, 120:176401, Apr 2018.

[109] Andrew W Long and Andrew L Ferguson. Nonlinear machine learning of patchy colloid self-assembly pathways and mechanisms. *The Journal of Physical Chemistry B*, 118(15):4228–4244, 2014.

[110] Xingkun Man, Shuang Yang, Dadong Yan, and An-Chang Shi. Adsorption and depletion of polyelectrolytes in charged cylindrical system within self-consistent field theory. *Macromolecules*, 41(14):5451–5456, 2008.

[111] M. Matsen and M. Schick. Stable and unstable phases of a diblock copolymer melt. *Phys. Rev. Lett.*, 72:2660–2663, Apr 1994.

[112] M. W. Matsen. Phase behavior of block copolymer/homopolymer blends. *Macromolecules*, 28(17):5765–5773, 1995.

[113] M. W. Matsen. The standard gaussian model for block copolymer melts. *Journal of Physics: Condensed Matter*, 14(2):R21, 2002.

[114] M. W. Matsen. Fast and accurate scft calculations for periodic block-copolymer morphologies using the spectral method with anderson mixing. *The European Physical Journal E*, 30(4):361, Dec 2009.

[115] M. W. Matsen and F. S. Bates. Unifying weak- and strong-segregation block copolymer theories. *Macromolecules*, 29(4):1091–1098, 1996.

[116] Mark W Matsen. Self-consistent field theory and its applications. *Soft Matter*, 1:87–178, 2006.

[117] Pankaj Mehta, Marin Bukov, Ching-Hao Wang, Alexandre GR Day, Clint Richardson, Charles K Fisher, and David J Schwab. A high-bias, low-variance introduction to machine learning for physicists. *arXiv preprint arXiv:1803.08823*, 2018.

[118] Pankaj Mehta and David J Schwab. An exact mapping between the variational renormalization group and deep learning. *arXiv preprint arXiv:1410.3831*, 2014.

[119] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[120] Ryszard S Michalski, Jaime G Carbonell, and Tom M Mitchell. *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013.

[121] Marcus Müller and Friederike Schmid. Incorporating fluctuations and dynamics in self-consistent field theories for polymer blends. In Christian Holm and Kurt Kremer, editors, *Advanced Computer Simulation Approaches for Soft Matter Sciences II*, pages 1–58. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[122] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[123] Michael A Nielsen. Neural networks and deep learning. *URL: http://neuralnetworksanddeeplearning. com/.(visited: 01.11. 2014)*, 2015.

[124] Tomi Ohtsuki and Tomoki Ohtsuki. Deep learning the quantum phase transitions in random electron systems: Applications to three dimensions. *J. Phys. Soc. Jpn.*, 86(4):044708, 2017.

[125] Lars Onsager. Crystal statistics. i. a two-dimensional model with an order-disorder transition. *Physical Review*, 65(3-4):117, 1944.

[126] Christos H. Papadimitriou. Computational complexity. In *Encyclopedia of Computer Science*, pages 260–265. John Wiley and Sons Ltd., 2003.

[127] Rajesh Parekh, Jihoon Yang, and Vasant Honavar. Constructive neural-network learning algorithms for pattern classification. *IEEE Trans. Neural Netw.*, 11(2):436–451, 2000.

[128] Karl Pearson. On lines and planes of closest fit to systems of points in space. *Philos. Mag.*, 2:559–572, 1901.

[129] Pedro Ponte and Roger G. Melko. Kernel methods for interpretable machine learning of order parameters. *Phys. Rev. B*, 96:205146, Nov 2017.

[130] Yuri O. Popov, Jonghoon Lee, and Glenn H. Fredrickson. Field-theoretic simulations of polyelectrolyte complexation. *Journal of Polymer Science Part B: Polymer Physics*, 45(24):3223–3230, 2007.

[131] Nataliya Portman and Isaac Tamblyn. Sampling algorithms for validation of supervised learning models for ising-like systems. 2016.

[132] William H Press. *Numerical recipes 3rd edition: The art of scientific computing.* Cambridge university press, 2007.

[133] A Kai Qin, Vicky Ling Huang, and Ponnuthurai N Suganthan. Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE transactions on Evolutionary Computation*, 13(2):398–417, 2009.

[134] Marcelino Quito, Christopher Monterola, and Caesar Saloma. Solving N-body problems with neural networks. *Phys. Rev. Lett.*, 86:4741–4744, May 2001.

[135] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

[136] Wen-Jia Rao, Zhenyu Li, Qiong Zhu, Mingxing Luo, and Xin Wan. Identifying product order with restricted boltzmann machines. *Phys. Rev. B*, 97(9):094207, 2018.

[137] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pages 63–71. Springer, 2004.

[138] Prince E Rouse Jr. A theory of the linear viscoelastic properties of dilute solutions of coiling polymers. *The Journal of Chemical Physics*, 21(7):1272–1280, 1953.

[139] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[140] Samuel H. Rudy, Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3(4), 2017.

[141] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533, 10 1986.

[142] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 44(1.2):206–226, Jan 2000.

[143] Frank Schindler, Nicolas Regnault, and Titus Neupert. Probing many-body localization with neural networks. *Phys. Rev. B*, 95(24):245134, 2017.

[144] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Net*, 61:85–117, 2015.

[145] Stefan Schnabel, Michael Bachmann, and Wolfhard Janke. Elastic lennard-jones polymers meet clusters: Differences and similarities. *The Journal of chemical physics*, 131(12):124904, 2009.

[146] Stefan Schnabel, Daniel T. Seaton, David P. Landau, and Michael Bachmann. Microcanonical entropy inflection points: Key to systematic understanding of transitions in finite systems. *Phys. Rev. E*, 84:011127, Jul 2011.

[147] Stefan Schnabel, Thomas Vogel, Michael Bachmann, and Wolfhard Janke. Surface effects in the crystallization process of elastic flexible polymers. *Chem. Phys. Lett.*, 476(46):201 – 204, 2009.

[148] DT Seaton, T Wüst, and DP Landau. Collapse transitions in a flexible homopolymer chain: Application of the wang-landau algorithm. *Phys. Rev. E*, 81(1):011802, 2010.

[149] Irwin H Segel. *Enzyme kinetics: behavior and analysis of rapid equilibrium and steady state enzyme systems*. Wiley New York, 1993.

[150] H. S. Seung, H. Sompolinsky, and N. Tishby. Statistical mechanics of learning from examples. *Phys. Rev. A*, 45:6056–6091, Apr 1992.

[151] Christopher J Shallue and Andrew Vanderburg. Identifying exoplanets with deep learning: A five planet resonant chain around kepler-80 and an eighth planet around kepler-90. *Accepted for publication in the The Astronomical Journal*.

[152] Tristan A. Sharp, Spencer L. Thomas, Ekin D. Cubuk, Samuel S. Schoenholz, David J. Srolovitz, and Andrea J. Liu. Machine learning determination of atomic dynamics at grain boundaries. *Proceedings of the National Academy of Sciences*, 115(43):10943–10947, 2018.

[153] An-Chang Shi. Self-consistent field theory. In Shiro Kobayashi and Klaus Müllen, editors, *Encyclopedia of Polymeric Nanomaterials*, pages 2199–2203. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[154] An-Chang Shi, Jaan Noolandi, and Rashmi C. Desai. Theory of anisotropic fluctuations in ordered block copolymer phases. *Macromolecules*, 29(20):6487–6504, 1996.

[155] B. W. Silverman and M. C. Jones. E. fix and j.l. hodges (1951): An important contribution to nonparametric discriminant analysis and density estimation: Commentary on fix and hodges (1951). *International Statistical Review*, 57(3):233–238, 1989.

[156] Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018.

[157] John C. Snyder, Matthias Rupp, Katja Hansen, Klaus-Robert Müller, and Kieron Burke. Finding density functionals with machine learning. *Phys. Rev. Lett.*, 108:253002, Jun 2012.

[158] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January 2014.

[159] H. Staudinger and Jules Meyer. Über neue organische phosphorverbindungen iii. phosphinmethylenderivate und phosphinimine. *Helvetica Chimica Acta*, 2(1):635–646, 1919.

[160] Paul J. Steinhardt, David R. Nelson, and Marco Ronchetti. Bond-orientational order in liquids and glasses. *Phys. Rev. B*, 28:784–805, Jul 1983.

[161] Gert R Strobl. *The physics of polymers*, volume 2. Springer, 1997.

[162] Shao-Tang Sun, Izumi Nishio, Gerald Swislow, and Toyoichi Tanaka. The coil–globule transition: Radius of gyration of polystyrene in cyclohexane. *The Journal of Chemical Physics*, 73(12):5971–5975, 1980.

[163] J.A.K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, Jun 1999.

[164] Gerald Swislow, Shao-Tang Sun, Izumi Nishio, and Toyoichi Tanaka. Coil-globule phase transition in a single polystyrene chain in cyclohexane. *Physical Review Letters*, 44(12):796, 1980.

[165] R. B. Thompson, K. O/. Rasmussen, and T. Lookman. Improved convergence in block copolymer self-consistent field theory by anderson mixing. *The Journal of Chemical Physics*, 120(1):31–34, 2004.

[166] Giacomo Torlai and Roger G. Melko. Learning thermodynamics with boltzmann machines. *Phys. Rev. B*, 94:165134, Oct 2016.

[167] Lloyd N Trefethen. *Spectral methods in MATLAB*, volume 10. Siam, 2000.

[168] Peter J. M. van Laarhoven and Emile H. L. Aarts. *Simulated annealing*, pages 7–15. Springer Netherlands, Dordrecht, 1987.

[169] B. Ph. van Milligen, V. Tribaldos, and J. A. Jiménez. Neural network differential equation and plasma equilibrium solver. *Phys. Rev. Lett.*, 75:3594–3597, Nov 1995.

[170] Evert P. L. van Nieuwenburg, Ye-Hua Liu, and Sebastian D. Huber. Learning phase transitions by confusion. *Nature Physics*, 13(5):435, 2017.

[171] Michael Walters, Qianshi Wei, and Jeff Z. Y. Chen. Machine learning topological defects of confined liquid crystals in two dimensions. *Phys. Rev. E*, 99:062701, Jun 2019.

[172] Ce Wang and Hui Zhai. Machine learning of frustrated classical spin models. i. principal component analysis. *Phys. Rev. B*, 96(14):144432, 2017.

[173] Fugao Wang and D. P Landau. Determining the density of states for classical statistical models: A random walk algorithm to produce a flat histogram. *Phys. Rev. E*, 64(5):056101, 2001.

[174] Fugao Wang and D. P. Landau. Efficient, multiple-range random walk algorithm to calculate the density of states. *Phys. Rev. Lett.*, 86:2050–2053, Mar 2001.

[175] Lei Wang. Discovering phase transitions with unsupervised learning. *Phys. Rev. B*, 94:195105, Nov 2016.

[176] Qiang Wang, Takashi Taniguchi, and Glenn H. Fredrickson. Self-consistent field theory of polyelectrolyte systems. *The Journal of Physical Chemistry B*, 108(21):6733–6744, 2004.

[177] Rui Wang and Zhen-Gang Wang. Theory of polymers in poor solvent: phase equilibrium and nucleation behavior. *Macromolecules*, 45(15):6266–6271, 2012.

[178] Zhen-Gang Wang. 50th anniversary perspective: Polymer conformation–a pedagogical review. *Macromolecules*, 50(23):9073–9114, 2017.

[179] Timothy L. H. Watkin, Albrecht Rau, and Michael Biehl. The statistical mechanics of learning a rule. *Rev. Mod. Phys.*, 65:499–556, Apr 1993.

[180] Jennifer N. Wei, David Duvenaud, and Alan Aspuru-Guzik. Neural networks for the prediction of organic chemistry reactions. *ACS Central Science*, 2(10):725–732, 2016. PMID: 27800555.

[181] Qianshi Wei, Ying Jiang, and Jeff Z. Y. Chen. Machine-learning solver for modified diffusion equations. *Phys. Rev. E*, 98:053304, Nov 2018.

[182] Qianshi Wei, Roger G. Melko, and Jeff Z. Y. Chen. Identifying polymer states by machine learning. *Phys. Rev. E*, 95:032504, Mar 2017.

[183] Xiao-Gang Wen. *Quantum field theory of many-body systems: from the origin of sound to an origin of light and electrons.* Oxford University Press on Demand, 2004.

[184] Sebastian J Wetzel. Unsupervised learning of phase transitions: From principal component analysis to variational autoencoders. *Phys. Rev. E*, 96(2):022140, 2017.

[185] George M. Whitesides and Bartosz Grzybowski. Self-assembly at all scales. *Science*, 295(5564):2418–2421, 2002.

[186] M. D. Whitmore and J. D. Vavasour. Self-consistent field theory of block copolymers and block copolymer blends. *Acta Polymerica*, 46(5):341–360, 1995.

[187] Ian H Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, 2005.

[188] Chi Wu and Xiaohui Wang. Globule-to-coil transition of a single homopolymer chain in solution. *Physical review letters*, 80(18):4092, 1998.

[189] Xin Xu, Qianshi Wei, Huaping Li, Yuzhang Wang, Yuguo Chen, and Ying Jiang. Recognition of polymer configurations by unsupervised learning. *Phys. Rev. E*, 99:043307, Apr 2019.

[190] Julia M Yeomans. *Statistical mechanics of phase transitions.* Clarendon Press, 1992.

[191] Bin Yu, Pingchuan Sun, Tiehong Chen, Qinghua Jin, Datong Ding, Baohui Li, and An-Chang Shi. Confinement-induced novel morphologies of block copolymers. *Phys. Rev. Lett.*, 96:138306, Apr 2006.

[192] Wancheng Yu, Yuan Liu, Yuguo Chen, Ying Jiang, and Jeff ZY Chen. Generating the conformational properties of a polymer by the restricted boltzmann machine. *The Journal of chemical physics*, 151(3):031101, 2019.

[193] Lixin Zhan, Jeff Z. Y. Chen, and Wing-Ki Liu. Determination of structural transitions of atomic clusters from local and global bond orientational order parameters. *J. Chem. Phys.*, 127(14):141101, 2007.

[194] Pengfei Zhang, Huitao Shen, and Hui Zhai. Machine learning topological invariants with neural networks. *Phys. Rev. Lett.*, 120(6):066401, 2018.

[195] Yi Zhang and Eun-Ah Kim. Quantum loop topography for machine learning. *Phys. Rev. Lett.*, 118:216401, May 2017.

[196] Yi Zhang, Roger G Melko, and Eun-Ah Kim. Machine learning z2 quantum spin liquids with quasiparticle statistics. *Phys. Rev. B*, 96(24):245119, 2017.

[197] Bruno H Zimm. Dynamics of polymer molecules in dilute solution: viscoelasticity, flow birefringence and dielectric loss. *The journal of chemical physics*, 24(2):269–278, 1956.