Deep Learning and Self-Consistent Field Theory: A Path Towards Accelerating Polymer Phase Discovery

Yao Xuan[1*], Kris T. Delaney[2], Hector D. Ceniceros[1], and Glenn H. Fredrickson[2,3]

[1]*Department of Mathematics, University of California, Santa Barbara, California 93106, USA*
[2]*Materials Research Laboratory, University of California, Santa Barbara, California 93106, USA*
[3]*Departments of Materials and Chemical Engineering, University of California, Santa Barbara, California 93106, USA*

Aug 2020

Abstract

A new framework that leverages data obtained from self-consistent field theory (SCFT) simulations with deep learning to accelerate the exploration of parameter space for block copolymers is presented. Deep neural networks are adapted and trained in Sobolev space to better capture the saddle point nature of the SCFT approximation. The proposed approach consists of two main problems: 1) the learning of an approximation to the effective Hamiltonian as a function of the average monomer density fields and the relevant physical parameters and 2) the prediction of saddle density fields given the polymer parameters. There is an additional challenge: the effective Hamiltonian has to be invariant under shifts (and rotations in 2D and 3D). A data-enhancing approach and an appropriate regularization are introduced to effectively achieve said invariance. In this first study, the focus is on one-dimensional (in physical space) systems to allow for a thorough exploration and development of the proposed methodology.

Keywords: Self-Consistent Field Theory; Machine Learning; Sobolev Space; Saddle Density Fields; Global Shift-invariance.

1 Introduction

Numerical simulations using self-consistent field theory (SCFT) are a valuable method to study the energetics and structure of polymer phases [1, 2, 3]. However, these computations are generally expensive. Relaxation methods to the SCFT (saddle point) solutions are slow to converge and

*corresponding author, yxuan@math.ucsb.edu

each iteration is costly as it requires the solution of one or several modified diffusion (Fokker-Planck) equations [1, 4, 5]. There are some recent attempts to combine SCFT with machine learning. Nakamura [6] proposed to predict polymer phase types by a neural network with a theory-embedded layer that captures the characteristic features of the phase via coarse-grained mean-field theory. Wei, Jiang, and Shi [7] introduced a neural network approach as a solver to the SCFT modified diffusion equations to accelerate the computation of the mean fields. In this work we leverage techniques of machine learning to obtain fast and accurate predictions of SCFT solutions after a suitable map from parameter space and average monomer density is learned. This new approach, which merges computer-simulated data with supervised learning, works for a large range of parameters after one single training and has the potential to dramatically accelerate the discovery and study of new polymer phases.

To describe the proposed approach in more detail, let us consider the simple model of an incompressible AB diblock copolymer melt[1]. The relevant parameters are χN (χ is the Flory parameter and N is the copolymer degree of polymerization), which measures the strength of segregation of the two components, L, the cell length in units of the unperturbed radius of gyration R_g, and f, the volume fraction of component A. Let $\rho = \rho_A$ be the local average monomer density of blocks A. Then, the framework we propose consists of solving the following two problems:

- **Problem 1**: Learn a map $(\chi N, L, f, \rho) \longmapsto \tilde{H}(\chi N, L, f, \rho)$, where $\tilde{H}$ is an accurate approximation of the field-theoretic intensive Hamiltonian H, the Helmholtz free-energy per chain at saddle points.

- **Problem 2**: For specific values χN^*, L^*, f^*, find accurately and efficiently the density field ρ^* that minimizes $\tilde{H}$.

Once $\tilde{H}$, a surrogate for the effective Hamiltonian at the saddle points, is learned (Problem 1), the procedure to solve Problem 2 can be expediently applied to screen the parameter space for new phase candidates.

To solve the first problem we recast H as

$$H(\chi N, L, f, \rho) = \chi N f - \frac{\chi N}{L^d} \int \rho^2 dr + R(\chi N, L, f, \rho), \tag{1}$$

by extracting the leading (quadratic) interaction term, and focus on learning, via neural networks, the remainder or residual term R, which contributes to polymer entropy, but not enthalpy. In Eq. (1), d is the spatial dimension. The rationale for splitting H is simply that we know the exact expression for the enthalpic part of H, and it is both local and economical to compute while the entropic part is not known in closed form and is computationally expensive. We present numerical results in Appendix 6.2 that show learning only the entropic part R produces superior results for the predicted density field ρ^* than those obtained by learning the full functional H.

Note that in an auxiliary field theory the effective Hamiltonian H, which coincides with the free energy at the SCFT saddle point, is a functional of one or more potential-chemical-like fields and one pressure-like field. The ρ dependence is generally integrated out using Gaussian integral identities and ρ is instead evaluated from these auxiliary fields. Here, we seek instead to learn directly an approximation of H, or more precisely of R, as a function of ρ.

Both Problem 1 and Problem 2 are challenging for the following reasons. First, the training data set for machine learning is generated from SCFT simulations, which produce physically meaningful information only at the saddle point. Thus, we only know H at the saddle point solutions, a reduced

[1]This SCFT model is fully described in Appendix 6.1.

set of a much larger complex manifold. But ultimately, we would like to find a minimizer of H over all possible ρ fields, including both saddle points and non-saddle points. Therefore, additional assumptions about the map are necessary. Second, the input is very high dimensional; the density field ρ becomes a large vector whose components correspond to values of ρ at regular mesh points in physical space. Consequently, even with a successfully learned map in Problem 1, Problem 2 is still a formidable optimization problem due to its high dimensionality. Moreover, H can have many local minima, thus finding a global minimum is a difficult task. Finally, the learned map $\tilde{H}$ needs to be invariant under translations (and rotations in 2D and 3D) and this poses an additional challenge to the map-learning process.

In this work, we propose to learn a map from $(\chi N, L, f, \rho)$ to the free energy, using SCFT-generated data, which is a good approximation of H and its gradient around saddle points and is invariant under spatial shifts of the density field. This is achieved by constructing a novel deep neural network in Sobolev space. Then, we pre-screen candidates for minimizers of the learned map to find, via gradient descent, predictors of new SCFT saddle points. In this first study, we focus on one-dimensional (in physical space) systems to allow for a thorough exploration and development of the proposed methodology.

The rest of this article is organized as follows. In Section 2, we propose the neural network-based method to approximate H in Sobolev space. By using a Sobolev metric we are able to predict both H and its gradient (thermodynamic force) simultaneously. Additional properties are added to the neural network based on the periodic shift invariance of the modeled system and the mathematical existence of the desired neural network is rigorously proved. Section 4 is devoted to a summary of numerical results for an AB diblock system and for an AB_3 mikto-arm star system, both with density-field variations in only one space dimension. Finally, some concluding remarks, including the prospects for extension to higher spatial dimensions, are given in Section 5.

2 The Neural Network Model

We describe in this section the deep neural network trained in Sobolev space that we propose to accurately describe H and its gradient in the vicinity of field-theoretic saddle points.

2.1 Methodology

Henceforth, we use $x = (\chi N, L, f, \rho)$ to denote all parameters of the model, which for concreteness is an incompressible diblock copolymer melt. Here, ρ is a vector whose components are the values of the local average density field ρ at the spatial grid points. We need to learn a map $(\chi N, L, f, \rho) \longmapsto \tilde{H}(\chi N, L, f, \rho)$ which approximates the effective Hamiltonian H of the field theory at the saddle points.

Suppose $M = \{(x_1, H_1), (x_2, H_2), ..., (x_{N_T}, H_{N_T})\}$ is the training set of size N_T generated by SCFT simulations, where $x_i = (\chi N_i, L_i, f_i, \rho_i)$ represents the i_{th} training point and H_i is the corresponding H value (free energy) for the i_{th} training point.

As mentioned in the introduction, we start by rewriting H as

$$H(x) = \chi N f - \frac{\chi N}{L} \int \rho^2 dr + R(x), \tag{2}$$

and learn the entropic remainder function R. An appropriate model for this task is one that matches both the functional value of R and its gradient (first variation in the continuum case) because at a saddle point ρ^*, necessarily $\frac{\delta H}{\delta \rho}|_{\rho=\rho^*} = 0$. We will call NN the approximation to R obtained via

a neural network approach that we specify in detail below. Hence, at the end, the approximate effective Hamiltonian map has the expression

$$\tilde{H}(x) = \chi N f - \frac{\chi N}{L} \int \rho^2 dr + NN(x). \tag{3}$$

In a neural network, there are many parameters in the form of weights and biases. These parameters are determined by minimizing a cost or loss function, which is defined based on the desired properties of the neural network. For our problem, we first choose a basic version of cost function as

$$C(\alpha) = \sum_{i=1}^{N_T} (\tilde{H}(x_i) - H_i)^2 + \beta \sum_{i=1}^{N_T} \|\nabla_\rho \tilde{H}(x_i)\|^2, \tag{4}$$

where α refers to the all the parameters in the neural network (hidden in the structure of $\tilde{H}$) and β is a parameter that controls the size of the penalty term $\sum_{i=1}^{N_T} \|\nabla_\rho \tilde{H}(x_i)\|^2$, which favors vanishing gradients at training points and adds smoothness to the learned map and prevents overfitting. More importantly, this regularizing term effectively enforces that the gradient of the learned $\tilde{H}$ approximates the zero vector at the training points (all training points are saddle points of H). It is important to emphasize that the learned map will ultimately match accurately both the function value of H and its vanishing gradient at training points. This property of the model represents a new approach to predict SCFT saddle points.

Note that for each data point (x_i, H_i), in view of (2) and (3), NN should match

$$R_i = H_i - \chi N_i f_i + \frac{\chi N_i}{L_i} \int \rho_i^2 dr. \tag{5}$$

Moreover, after discretizing in space we have

$$\nabla_\rho \tilde{H}(x_i) = -\frac{2\chi N_i}{L_i} \Delta r \, \rho_i + \nabla_\rho NN(x_i), \tag{6}$$

where Δr is the spatial mesh size. Then, we can rewrite the cost function (4) in terms of the residual map NN as

$$C(\alpha) = \sum_{i=1}^{N_T} (NN(x_i) - R_i)^2 + \beta \sum_{i=1}^{N_T} \|\nabla_\rho NN(x_i) - \frac{2\chi N_i}{L_i} \Delta r \, \rho_i\|^2. \tag{7}$$

From Eq. (7), the cost function we seek to minimize is just the distance between the predicted map, NN, and the actual map we are trying to approximate, R, in the sense of the Sobolev norm. The existence of a neural network to approximate a map to a desired accuracy has been established for different activation functions [8, 9]. Thus, a natural question is: does there exist a neural network that approximates both the map and its gradient, i.e. in the sense of Sobolev norm? The answer is yes and our approach is built on this solid theoretical foundation.

A single hidden layer feedback network constructs functions of the form

$$g(x) = \sum_{j=1}^{q} \omega_j \, G(\theta_j \cdot \tilde{x}), \tag{8}$$

where G is an activation function (e.g. sigmoid, softmax, ReLU, etc.), $\tilde{x} = (1, x)$, ω_j represents hidden-to-output layer weights and the vectors θ_j (of dimension equal to the dimension of x plus 1)

represent input-to-hidden layer weights. Collectively, these parameters are the vector α appearing in the cost or loss function (7). If this class of functions is dense in a functional space F under some specific metric, then for any $f \in F$, there is a neural network of the form (8) that approximates f to a given accuracy in that metric. Hornik, Stinchcombe, and White [10] proved this is the case with the Sobolev norm. This result guarantees the existence of a single-hidden-layer neural network that approximates both a functional and its functional gradient simultaneously and provides the theoretical foundation for our proposed cost function (7). Moreover, there also exists, as proved by Czarnecki, Osindero, Jaderberg, Swirszcz, and Pascanu [11], a single layer neural network that matches both the functional and its functional gradient with *zero training loss*. That is, theoretically, there exists α^* such that $C(\alpha^*) = 0$. A more technical summary of these important theorems is presented in the Appendix 6.3.

We use stochastic gradient descent methods, employing back propagation and the Adam method, to find the network's parameters by minimizing (7). We designed a deep neural network with 6 hidden layers to learn NN; the architecture of this deep neural network is shown in Table 1. The width and depth of neural network, which control the generalization power and convergence rate, are tunable hyperparameters. We selected these hyperparameters guided by an ablation study, which is summarized in Appendix 6.5.

Hidden layer	1	2	3	4	5	6
Number of neurons	60	180	180	180	180	60

Table 1: Architecture of the deep neural network: number of neurons (cells) in each hidden layer.

2.2 Neural Network with Global Shift-Invariance

In the SCFT model, with periodic boundary conditions, the effective Hamiltonian is invariant under shifts in ρ. Hence, our approximation $\tilde{H}$ should have the property

$$\tilde{H}(x) = \tilde{H}(T_s x), \tag{9}$$

where $T_s x = (\chi N, L, f, T_s\rho)$ and $T_s\rho$ is a spatial shift of the periodic density field ρ, i.e., $T_s\rho(r)$=$\rho(r+s)$. Equivalently, $\tilde{H}(\chi_N, L, f, [\rho_1, \rho_2, ...]) = \tilde{H}(\chi_N, L, f, [\rho_{1+s}, \rho_{2+s}, ...\rho_1, ..., \rho_s])$. This is illustrated in Figure 1: any period-window of ρ generates the same effective Hamiltonian. This is of course also the case for the entropic remainder term R. Machine learning techniques with global shift-invariance are not yet common in the literature. Some popular deep learning architectures, such as the convolutional neural network (CNN), have *local* shift-invariance. However, it is easy to construct examples in which a small change in input can result in a huge change of the output in a CNN [12]. Our goal here is to construct a deep neural network with global shift-invariance. This can be achieved by combining data augmentation and the addition of a penalty or regularization term in the loss function. Data augmentation consists in adding (cyclically) shifted data points to training set so that the neural network can learn the pattern of periodic shift-invariance directly from the data.

Suppose $M = \{(x_i, H_i), 1 \leq i \leq N_T\}$ is the original training set, the new training set after augmentation is defined as:

$$\tilde{M} = \{(T_s x_i, H_i), 1 \leq i \leq N_T, 1 \leq s \leq N_s\},$$

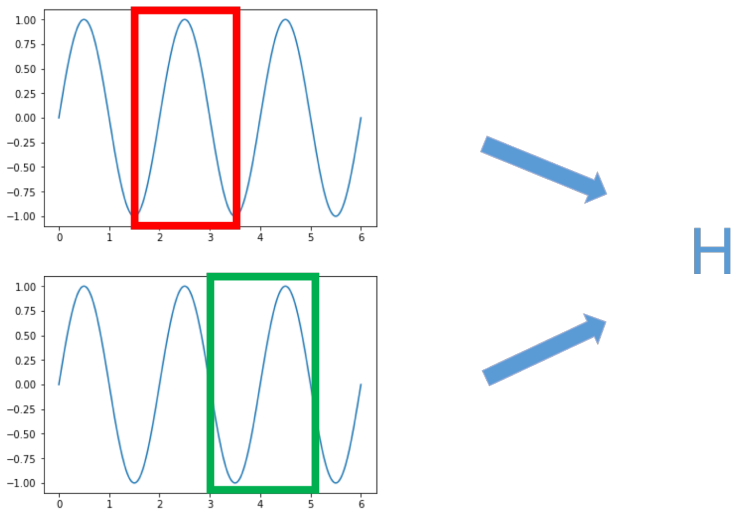


Figure 1: Density field in any period window generates the same effective Hamiltonian. This global shift-invariance is required for the learned map.

where N_s is number of possible shifts. We also modify the cost or loss function with an additional penalty term as follows

$$\begin{aligned} C(\alpha) = &\sum_{s=1}^{N_s}\sum_{i=1}^{N_T}(NN(T_s x_i) - \tilde{H}_i)^2 \\ &+ \beta \sum_{s=1}^{N_s}\sum_{i=1}^{N_T} \|\nabla_\rho NN(T_s x_i) - \frac{2\chi N_i}{L_i}\Delta r T_s \rho_i\|^2 \\ &+ \gamma \sum_{s=1}^{N_s}\sum_{i=1}^{N_T}(NN(x_i) - NN(T_s x_i))^2. \end{aligned} \tag{10}$$

The last term penalizes the differences of NN at shifted points. It is worth noting that while global shift-invariance is essential for many physical models, the machine learning literature on that topic is scarce. Global shift-invariance is a challenging problem because it results in a much smaller subspace of the general neural network approximation space. Therefore, the existence of a neural network, in the global shift-invariant space, that approximates accurately a functional in the Sobolev norm is an important and non-trivial question. Indeed, if the subspace is not large enough, the existence of a network to approximate accurately any given functional is not guaranteed. Fortunately, we are able to prove the existence of a neural network that approximates a given functional under the Sobolev norm and has global shift-invariance at the same time. In more precise terms, we have the following result.

Theorem 1. There exists a neural network $NN(x)$ with a ReLU (or a leaky ReLU) activation function such that the neural network has zero training loss with loss function (10).

Proof. Let $f(x) = R(x)$, i.e. the true remainder of the effective Hamiltonian and $g(\rho) = \frac{2\chi_N}{L}\Delta r \rho$, the true gradient of the remainder. Then, we can take any extension of $g(\rho)$, say $\tilde{g}(x)$, such that $\tilde{g}(x)$ matches $g(\rho)$ on all the coordinates corresponding to ρ. By Theorem 4 in the Appendix, there is a neural network such that $NN(x)$ matches $f(x)$ and $NN_x(x)$ matches $\tilde{g}(x)$ exactly, on

the training set. Since we only need the gradient with respect to ρ, by the definition of $\tilde{g}(x)$, NN_ρ is equal to $g(\rho)$ on the training set. This directly makes the first two terms vanish in cost function (10). For the third term, noting that

$$NN(T_s x_i) = R(T_s x_i) = R(x_i) = NN(x_i),$$

for all i, s, we get $NN(x_i) - NN(T_s x_i) = 0$. Thus, there exists a neural network $NN(x)$ that leads to a zero training loss with the cost function (10). In other words, it is possible to achieve simultaneously the desired approximation in the Sobolev norm and global shift-invariance with one neural network. □

2.3 Searching for Saddle Points

As described above, the map $\tilde{H}$ is trained in Sobolev space. Once this is achieved, we can use local minimizers of $\tilde{H}$, obtained via gradient descent, as predictors of the saddle point density fields of the effective Hamiltonian.[2] To solve $\nabla_\rho \tilde{H} = 0$ we employ the gradient descent method because it is simple and fast to evaluate the learner and its gradient at any x.

Given arbitrary $(\chi N^*, L^*, f^*)$, we eliminate these variables in $\tilde{H}(x)$ so that $\tilde{H}$ can be viewed as a function of ρ alone. Then, we proceed with gradient descent to find a local minimizer $\widehat{\rho}^{NN}$ as an approximation to a saddle point ρ^* of the effective field-theoretic Hamiltonian:

$$\rho^{n+1} = \rho^n - \epsilon \nabla_\rho \tilde{H}(x^n), \tag{11}$$

where $x^n = (\chi N^*, L^*, f^*, \rho^n)$ and ϵ is the step size. Note that, as expressed in (6),

$$\nabla_\rho \tilde{H}(x) = -\frac{2\chi_N}{L} \Delta r \, \rho + \nabla_\rho NN(x), \tag{12}$$

where $\nabla_\rho NN(x)$ can be efficiently evaluated by the Chain Rule applied to the neural network, e.g. $\nabla_\rho g(x) = \sum_{j=1}^{q} \omega_j \nabla_\rho G(\theta_j \cdot \tilde{x})$ for a single-layer neural network. This is easy to compute even for a multiple-layer neural network because a computation graph is automatically generated for the neural network to gather gradient information for backward propagation. Again, here Δr is the spatial mesh resolution. The gradient descent iteration (20) is terminated when $\|\nabla_\rho \tilde{H}\|$ decreases below a target specified small value..

One important component of the gradient descent method is the initial iterate ρ^0. Here we propose a selection strategy for ρ^0 that fully uses the information of the training set. For a given $(\chi N^*, L^*, f^*)$, we scan all the density fields in the training set to find the one that generates the smallest $\|\nabla_\rho \tilde{H}\|$ at $(\chi N^*, L^*, f^*)$. Since a neural network is just a composition of a series of linear functions and simple (nonlinear) activation functions, this evaluation is extremely fast in comparison with the evaluation of the full SCFT model.

2.4 Strategies to Tune Hyperparameters

Hyperparameter tuning is an important procedure in deep learning. The weights and biases in the network, denoted collectively as α above, are parameters. There are multiple hyperparameters in a neural network model, such as the strengths of the penalization terms β and γ in the loss function, the hyperparameters defining the architecture of the network, such as the number of layers and the number of nodes per layer, and the learning rate (step size) used in the training process.

[2]Henceforth, we call the local minimizer of $\tilde{H}$ a saddle point predictor or a predicted saddle point.

Following standard practice, we split the dataset into three parts: a training set, a validation set, and a test set. For various combinations of different values of hyperparameters, the neural network is trained on the training set and the results are compared with the data in the validation set. After selecting the optimal combination of hyperparameters based on the performance on the validation set, we evaluate the best model on the test set to check the error. This comparison requires the specification of a metric. For the given validation set with parameters $(\chi N_i^*, L_i^*, f_i^*)$ where $1 \leq i \leq N_V$, we define the metric or loss function on the validation set as follows

$$\begin{aligned} C_V(\alpha) = &\sum_{s=1}^{N_s}\sum_{i=1}^{N_V}(NN(T_s x_i^*) - R_i)^2 \\ &+ \beta_V \sum_{s=1}^{N_s}\sum_{i=1}^{N_V} \|\nabla_\rho NN(T_s x_i^*) - \frac{2\chi N_i^*}{L_i^*}\Delta r T_s \rho_i^*\|^2 \\ &+ \gamma_V \sum_{s=1}^{N_s}\sum_{i=1}^{N_V}(NN(x_i^*) - NN(T_s x_i^*))^2 \\ &+ \theta_V \sum_{i=1}^{N_V} \|\widehat{\rho}_i^{NN} - \rho_i^*\|^2, \end{aligned} \tag{13}$$

where $x_i^* = (\chi N_i^*, L_i^*, f_i^*, \rho_i^*)$ and $\widehat{\rho}_i^{NN}$ is the predicted density field corresponding to $(\chi N_i^*, L_i^*, f_i^*)$. The first two terms provide a measure of the accuracy of the learner and its gradient. The third term measures the shift-invariance. The fourth term measures the deviation of predicted minimizers $\widehat{\rho}^{NN}$ from the saddle points. Thus, with this metric on the validation set we can find hyperparameters that lead to a neural network that not only approximates the effective Hamiltonian and its gradient but also predicts the saddle point after gradient descent searching. In the implementation, we can adjust the weights (β_V, γ_V and θ_V) to each of the terms in C_V to prioritize the approximation of NN, its gradient, its shift-invariance, or a saddle point, local density field or to balance the scaling.

2.5 The Algorithm

In training deep networks it is common to add a regularization term to the loss function that penalizes the size of weights and biases and provides smoothness to the model. Here, we follow that practice and modify our loss function by adding the term $\lambda\|\alpha\|_2^2$, i.e.

$$\begin{aligned} C(\alpha) = &\sum_{s=1}^{N_s}\sum_{i=1}^{N_T}(NN(T_s x_i) - R_i)^2 \\ &+ \beta \sum_{s=1}^{N_s}\sum_{i=1}^{N_T} \|\nabla_\rho NN(T_s x_i) - \frac{2\chi N_i}{L_i}\Delta r T_s \rho_i\|^2 \\ &+ \gamma \sum_{s=1}^{N_s}\sum_{i=1}^{N_T}(NN(x_i) - NN(T_s x_i))^2 + \lambda\|\alpha\|_2^2. \end{aligned} \tag{14}$$

Algorithm 1 provides the sequence of steps for deep learning H and to obtain predictions of SCFT saddle points. Training of the neural network is done under the Sobolev norm to obtain accurate predictions of both H and its gradient. With this network, after selecting a good, data-based initial guess, the gradient descent method is used to find accurate approximations of the saddle point.

Algorithm 1: Neural network method to learn the effective Hamiltonian and to obtain a saddle point density field prediction.

Input: Training set M, validation set M_V, test set M_T

1. Hyperparameter tuning:

for *hyperparameter* λ_i, β_j, γ_k **do**

  use Adam method to search for $\alpha_{ijk} = \arg\min_\alpha C(\alpha)$

  evaluate $C_V(\alpha_{ijk})$ on M_V

end

$\lambda, \beta, \gamma = \arg\min_{\lambda_i,\beta_j,\gamma_k} C_V(\alpha_{ijk})$

2. Prediction of effective Hamiltonian on M_T:

for $x = (\chi N, L, f, \rho)$ **do**

  $\tilde{H}(x) = \chi N f - \frac{\chi N}{L} \int \rho^2 dr + NN(x)$

end

3. Prediction of saddle point corresponding to $\chi N^*, L^*, f^*$:

for ρ_i *from* M **do**

  Evaluate $\tilde{H}(\chi N^*, L^*, f^*, \rho_i)$, $\nabla_\rho \tilde{H}(\chi N^*, L^*, f^*, \rho_i)$

end

$\rho^0 = \arg\min_{\rho_i} \|\nabla_\rho \tilde{H}(\chi N^*, L^*, f^*, \rho_i)\|$

while $\|\nabla_\rho \tilde{H}(\chi N^*, L^*, f^*, \rho_i)\| > \delta$ **do**

  $\rho^{n+1} = \rho^n - \epsilon \nabla_\rho \tilde{H}(x^n)$

end

$\rho^n \to \tilde{\rho}^{NN}$ which is the estimated saddle point corresponding to $\chi N^*, L^*, f^*$

3 Exploring a Simpler Data-Based Learner

As we show in Section 4, our deep NN-based approach is remarkably accurate and efficient in building a shift-invariant surrogate for the field-theoretic effective Hamiltonian in the vicinity of saddle points. There is however a vast number of other data-based learners that one could potentially use within our proposed framework to accelerate the exploration of parameter space in polymer SCFT. In fact, the first learner we tried, inspired by the pioneering work of Snyder, Rupp, Hansen, Blooston, Müller, and Burke [13, 14, 15], was Kernel Ridge Regression (KRR).

In KRR, one constructs the approximate map $\tilde{H}$ from an expansion of the form

$$\tilde{H}(x) = \sum_{j=1}^{N_T} \alpha_j K(x, x_j), \tag{15}$$

where, as before, $x = (\chi N, L, f, \rho)$, x_i represents i_{th} training point, K is a fixed function known as the kernel (e.g. the Gaussian function), and N_T is the training set size. Because the number of parameters α_j is not fixed and depends on the size N_T of the data , KRR is a simple data-adaptive regression approach.

To use KRR as the data-based learner in our proposed, accelerated SCFT framework we need 1) to train KRR (i.e. determine $\alpha_1, \alpha_2, ..., \alpha_{N_T}$), in Sobolev space to approximate simultaneously

H and its gradient, and 2) to constrain the KRR learner to be shift-invariant. While we could solve 1) explicitly, we were unable to find a satisfactory solution for 2). Although this is a serious limitation of KRR in the context of polymer SCFT, the extension of the KRR to Sobolev space training could be useful for other applications that do not require the global invariance. With this in mind, we present below this extension and compare this approximation with that produced by the deep NN learner.

Specifically, to train KRR in Sobolev space and determine $\alpha_1, \alpha_2, ..., \alpha_{N_T}$, we minimize the cost function

$$C(\alpha) = \sum_{i=1}^{N_T} (\tilde{H}(x_i) - H_i)^2 + \beta \sum_{i=1}^{n} \|\nabla_\rho \tilde{H}(x_i)\|^2 + \lambda\, \alpha^T K \alpha, \tag{16}$$

where $\nabla_\rho \tilde{H}(x_i)$ stands for the gradient of $\tilde{H}$ with respect to ρ evaluated at x_i, $\alpha = (\alpha_1, \alpha_2, ..., \alpha_{N_T})^T$, and K is a matrix with entries $K_{ij} = K(x_i, x_j)$. The regularization term $\lambda\, \alpha^T K \alpha$ penalizes the size of the coefficients α in the metric induced by the kernel and limits overfitting. The gradient term in $C(\alpha)$ favors approximations with a vanishing gradient, consistent with SCFT saddle points.

We now derive an explicit expression for α in our Sobolev space KRR model. We employ the Gaussian kernel

$$K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right), \tag{17}$$

where

$$\|x_i - x_j\| = \sqrt{c_1(\chi N_i - \chi N_j)^2 + c_2(L_i - L_j)^2 + c_3(f_i - f_j)^2 + \|\rho_i - \rho_j\|_2^2}, \tag{18}$$

$\|\cdot\|_2$ is the l_2 norm, and c_1, c_2, c_3 are positive hyperparameters.

Theorem 2. The Sobolev-trained Kernel Ridge Regression (15) with Gaussian kernel (17) , which minimizes (16), has coefficients α that satisfy linear system

$$(K^T K + \beta \tilde{K} + \lambda K)\alpha = K^T H, \tag{19}$$

where $\tilde{K} = \sum_{i=1}^{N_T} \tilde{K}_i$ and $\tilde{K}_i$ is a matrix with $(\tilde{K}_i)_{jm} = \frac{1}{\sigma^4} K(x_i, x_j) K(x_i, x_m) \langle \rho_j - \rho_i, \rho_m - \rho_i \rangle$.

A proof is given in Appendix 6.4. This result is an attractive feature of KRR; there is an explicit expression for α, i.e. the learner is easy to train.

There are 6 hyperparameters in our KRR method: c_1, c_2, c_3, λ, σ, and β. They could either be determined by the methods in Section 2.4 or by k-fold cross validation. Once the KRR learner is obtained, we proceed as in the deep neural network case, to minimize $\tilde{H}$, via gradient descent, to find saddle point predictors

$$\rho^{n+1} = \rho^n - \epsilon \nabla_\rho \tilde{H}(x^n), \tag{20}$$

where the gradient is evaluated explicitly:

$$\nabla_\rho \tilde{H}(x) = \sum_{j=1}^{N_T} \frac{\rho_j - \rho}{\sigma^2}\, \alpha_j\, K(x, x_j). \tag{21}$$

4 Results

We conducted several numerical experiments to validate the proposed SCFT-deep learning framework. We summarize our results in this section.

We considered two systems: an AB diblock copolymer melt and an AB_3 star copolymer melt, both in one spatial dimension. There were two stages in the numerical experiments. First, we trained a machine learning architecture (the deep neural network here, but other machine learning methods could be used in this step) to predict the effective Hamiltonian H and its gradient from the parameters, χN, L, f and average monomer (A) density field ρ. The deep neural network learner is equipped with global shift-invariance obtained through a data augmentation technique and the addition of a penalty term in the loss function. Second, we selected an initial guess for the density field from the training set and searched by gradient descent for a density field that minimizes the learned map $\tilde{H}$.

4.1 AB Diblock Copolymer with Low-to-Moderate χN

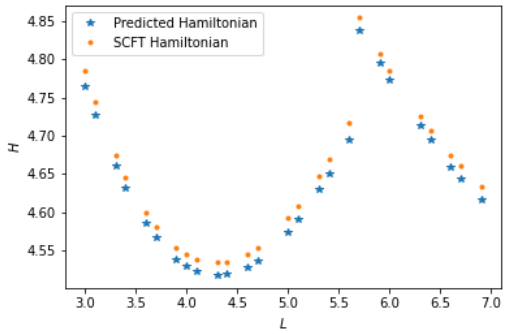
In this experiment, we first test the algorithms on an AB diblock system for χN in the range $[20, 35]$ for A-block volume fractions $f \in \{0.3, 0.4\}$. The training, validation, and test data for $f = 0.3$ are as follows:

- The **training set** consists of the combination of $\chi N \in \{n \in \mathbb{N} : 20 \leq n \leq 35\}$, $L \in \{n + 0.2, n + 0.5, n + 0.8 : 3 \leq n \leq 6\}$ and $f \in \{0.3\}$. The size of the training set is $16 * 3 * 4 = 192$.
- The **validation** and **test sets** consist of the combination of $\chi N \in \{n, n + 0.5 : 20 \leq n \leq 35, n \in \mathbb{N}\}$, $L \in \{n, n+0.1, n+0.3, n+0.4, n+0.6, n+0.7, n+0.9 : 3 \leq n \leq 6\}$ and $f \in \{0.3\}$. Note that this larger set has no overlap with the training set. We randomly take $192/3 = 64$ points by uniform distribution from this set as our validation set and apply the rest as the test set.

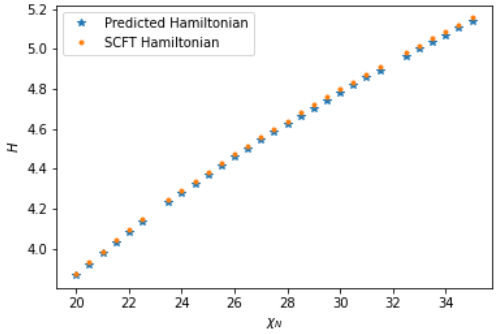
All the data points are obtained by numerically solving the SCFT model for each corresponding set of parameters. The modified diffusion equations were solved using periodic boundary conditions and pseudo-spectral collocation in space with 64 mesh points in 1D. 100 contour steps were made using second-order operator splitting [16, 17]. Auxiliary fields, initialized with smooth fields with a fixed number of periods, were relaxed to saddle-point configurations using the semi-implicit Seidel iteration [4]. L was varied rather than set to the value that minimizes the effective Hamiltonian in building the datasets. This gave more richness to the training set by including stressed configurations. However, as L increases, solutions with an increased number of periods become feasible; the algorithm selects the density field that produces the smallest effective Hamiltonian among these solutions with different periods. We use a similar strategy to generate the data for the case $f = 0.4$. The deep neural network is trained separately for $f = 0.3$ and $f = 0.4$.

Figure 2 shows the excellent accuracy of the predicted $\tilde{H}$ as a function of χN and L. $\tilde{H}$ coincides with the SCFT mean-field free energy H to two digits of accuracy. As mentioned above, global shift-invariance is an important property the SCFT model solutions. Figure 3a displays the output of the deep network for all the possible periodic shifts (cyclic permutations) of the density array input (from 0 to 63) for some representative cases. As this figure demonstrates, the proposed deep network preserves with good accuracy (2–3 digits) the desired, global shift-invariance.

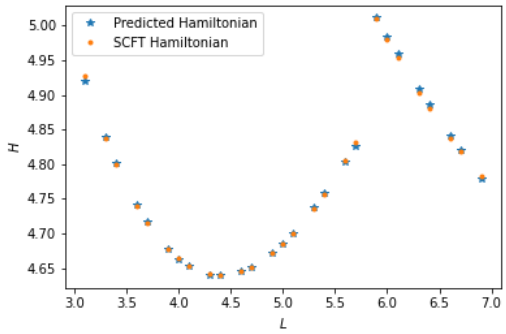
We now test the ability of our deep learning model to predict saddle point SCFT densities. As mentioned earlier this is done using gradient descent of the learned map. The search process



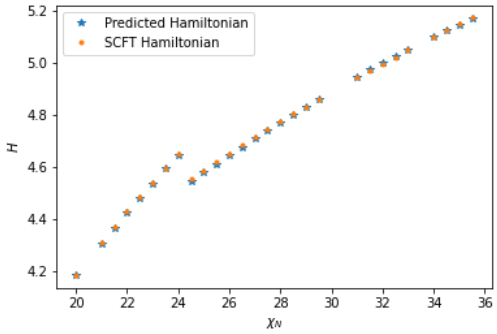
(a) $\chi N = 27.5$, $3 \leq L \leq 6.9$, and $f = 0.3$.



(b) $L = 3.6$, $20 \leq \chi N \leq 35.5$, and $f = 0.3$.

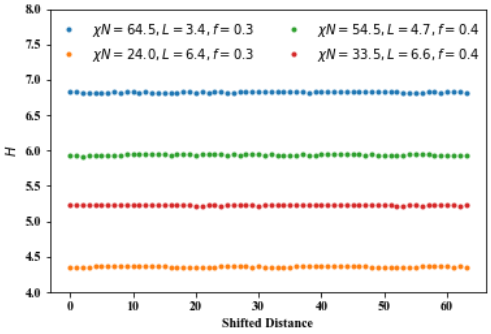


(c) $\chi N = 28.5$, $3 \leq L \leq 6.9$, and $f = 0.4$.

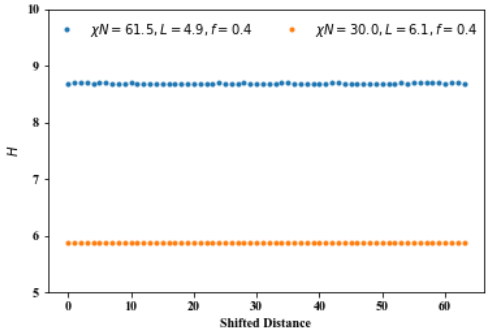


(d) $L = 5.6$, $20 \leq \chi N \leq 35.5$, and $f = 0.4$.

Figure 2: AB diblock copolymer with low-to-moderate χN: comparison of the deep learned effective Hamiltonian with the SCFT effective Hamiltonian. In (d), there is a jump because the optimal density field switches from two periods to one period at that point and this leads to a smaller effective Hamiltonian.



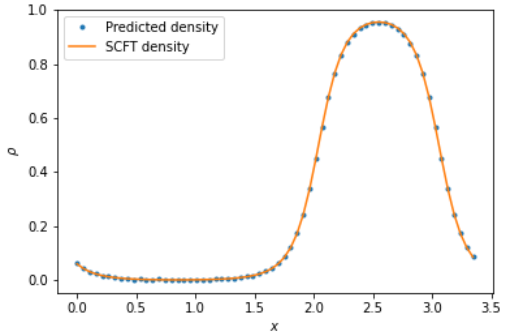
(a) AB diblock copolymer



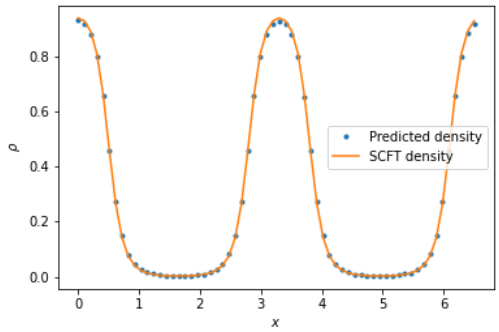
(b) AB_3 star copolymer

Figure 3: Global shift-invariance validation for some representative cases: (a)low-to-moderate χN and high χN cases for AB diblock system, (b)low-to moderate and high χN cases for AB_3 star system.

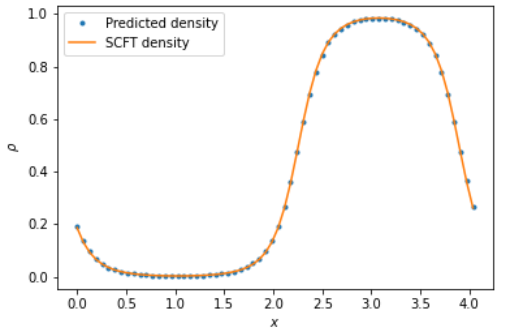
starts from the density field in the training set that generates a gradient of the learner $\tilde{H}$ closest to 0. Then, in each iteration, we evaluate the gradient of the learner by summing up the gradient of the leading order term and the gradient of the deep-learned remainder R, computed by the Chain Rule. Because the training is done in Sobolev space, a minimizer of the learner $\tilde{H}$ yields an accurate approximation of the true SCFT saddle point density field. Figure 4 presents four examples of the predicted density field and compares them with the corresponding SCFT solution. The accuracy of the deep learning model predictions is outstanding. Table 2 lists the specific errors of the predictions on the test set for both the AB diblock and the AB_3 star copolymer.



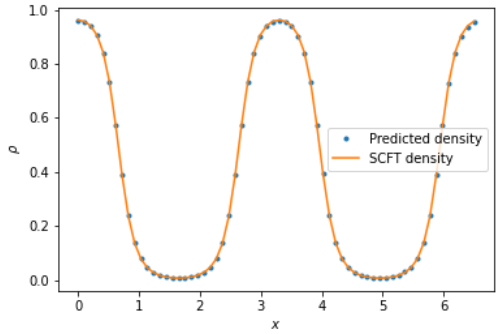
(a) $\chi N = 33.5$, $L = 3.4$, and $f = 0.3$.



(b) $\chi N = 30.5$, $L = 6.6$, and $f = 0.3$.



(c) $\chi N = 26.5$, $L = 4.1$, and $f = 0.4$.



(d) $\chi N = 24.0$, $L = 6.6$, and $f = 0.4$.

Figure 4: AB diblock copolymer with low-to-moderate χN: predicted and SCFT saddle density fields.

We are using a relatively small training set but get fairly accurate approximations. Increasing the number of data points would generally improve the performance. As proved by Chen, Jiang, Liao, and Zhao [18], there exists a deep ReLU architecture such that the mean squared error of the approximation converges at the rate of $O(n^{-\frac{2(s+\alpha)}{2(s+\alpha)+d}} \log^3 n)$ when n points are sampled to approximate a Hölder function in $\mathcal{H}^{s,\alpha}$ supported on a d-dimensional Riemannian manifold isometrically embedded in $\mathbb{R}^D$ with sub-Gaussian noise and a data intrinsic dimension of d.

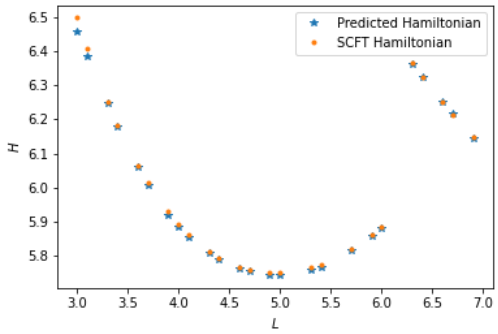
4.2 AB Diblock Copolymer with High χN

We now consider the case of high χN, i.e. strong segregation. This is a notoriously difficult case from the SCFT computational point of view. High χN SCFT simulations usually require high

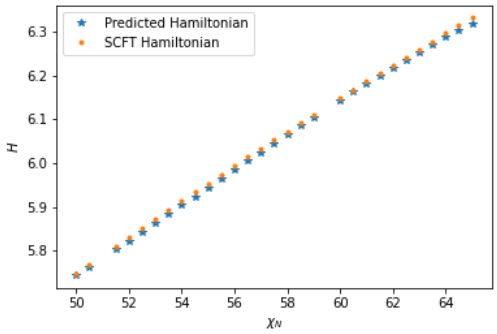
χN	Structure	f	E_H	E_G	E_I
low χN	AB diblock	0.3	0.017	0.018	0.006
		0.4	0.006	0.023	0.005
	AB_3 star	0.4	0.013	0.035	0.005
high χN	AB diblock	0.3	0.011	0.050	0.006
		0.4	0.012	0.046	0.017
	AB_3 star	0.4	0.017	0.065	0.012

Table 2: Performance of the Sobolev space-trained deep neural network on the test set (root mean square error): E_H is the error of predicted effective Hamiltonian, E_G is the error of predicted effective Hamiltonian gradient, E_I is the difference of predicted effective Hamiltonian of shifted density fields. Low χN refers to $\chi N \in [20, 35.5]$ for all cases. High χN refers to $\chi N \in [50, 65.5]$ for the AB diblock with $f = 0.3$ and for the AB_3 star system, and to $\chi N \in [45, 60.5]$ for the AB diblock with $f = 0.4$.

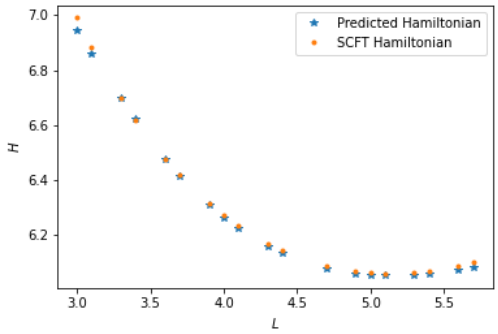
spatial resolution and are numerically stiff, taking hundreds or thousands of iterations to converge. Here, we select χN in the range $[50, 65]$ for $f = 0.3$ and χN in the range $[45, 60]$ for $f = 0.4$, using a similar procedure with same number of spatial and contour grid points as Section 4.1 to generate the dataset.



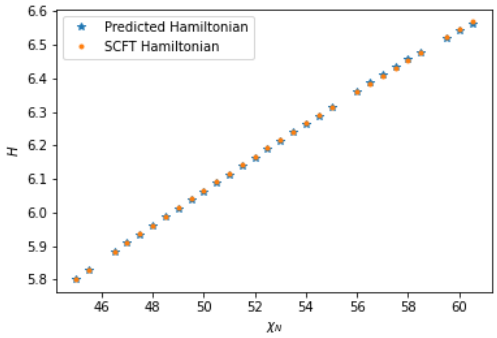
(a) $\chi N = 50.5$, $3 \leq L \leq 6.9$, and $f = 0.3$.



(b) $L = 4.6$, $50 \leq \chi N \leq 65.5$, and $f = 0.3$.



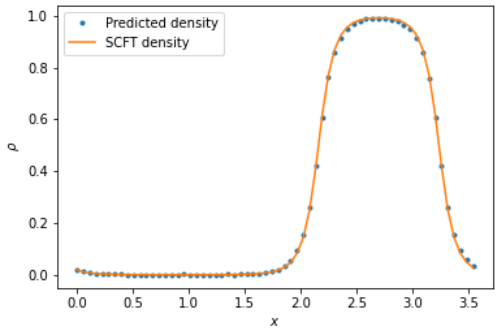
(c) $\chi N = 58.5$, $3 \leq L \leq 5.8$, and $f = 0.4$.



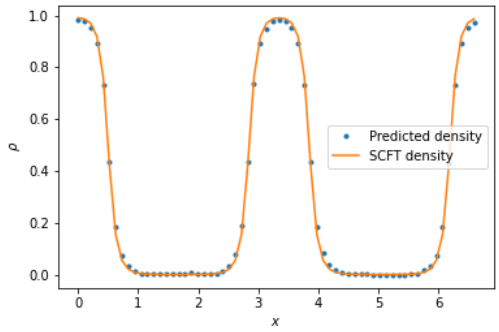
(d) $L = 3.6$, $45 \leq \chi N \leq 60.5$, and $f = 0.4$.

Figure 5: AB diblock copolymer with high χN: comparison of the learned and SCFT Hamiltonian.

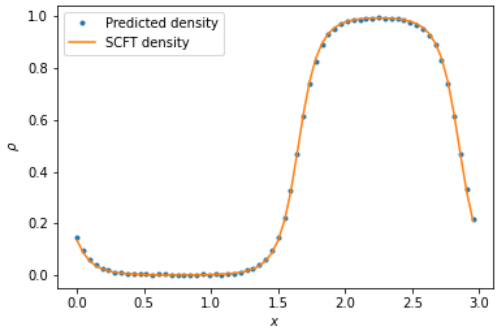
Figure 5 shows representative predictions of the effective Hamiltonian map H from $(\chi N, L, f, \rho)$, and Figure 3a displays representative results for global shift-invariance. As in the low-to-intermediate χN cases, the deep learner produces excellent predictions. Table 2 demonstrates that we still have several digits of accuracy in H, its gradient, and in preserving global shift-invariance. However, as expected, the error in the gradient is larger for the high χN cases because of the sharper interfaces.



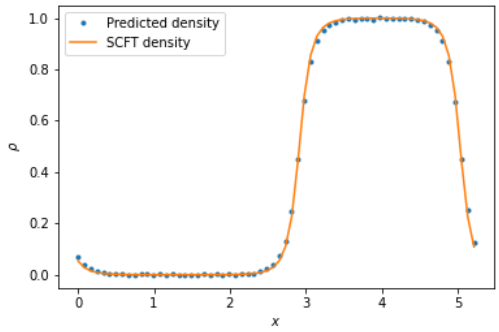
(a) $\chi N = 59.5$, $L = 3.6$, and $f = 0.3$.



(b) $\chi N = 60.0$, $L = 6.7$, and $f = 0.3$.



(c) $\chi N = 52.0$, $L = 3.0$, and $f = 0.4$.



(d) $\chi N = 57.5$, $L = 5.3$, and $f = 0.4$.

Figure 6: AB diblock copolymer with high χN: predicted and SCFT saddle density field.

Predictions of the SCFT saddle point density field for a given $(\chi N^*, L^*, f^*)$ are shown in Figure 6. The deep learner in combination with gradient descent is able to predict the density profiles for these higher values of χN almost as accurately as for the lower χN values. We emphasize that high χN SCFT computations are computationally much more expensive than those for smaller χN due to the requirement of higher numerical resolution and slower convergence of the saddle point iterations. In contrast, with the deep learning method we can obtain an accurate prediction of the saddle point density field for high χN fast, at the same cost as that for obtaining a density prediction for low-to-moderate χN case; in both cases, it is just several evaluations of a neural network, which is a combination of linear functions and activation functions as shown in Eq. (8). After generating hundreds of training data points and a one-time training, the Sobolev-trained neural network becomes a valuable, fast computational tool to predict accurately the effective Hamiltonian from density fields on any dataset, for example a large scale dataset with tens of thousands of data points to evaluate. No additional training or fine-tuning is needed during the subsequent polymer phase discovery process. To illustrate the running time difference between the

predictions by the numerical solution of SCFT and the new deep neural network model, we take 200 samples of $(\chi N, L, f)$ with $61 \leq \chi N \leq 65$, to compare both approaches. The running times are shown in Table 3. The SCFT CPU and the neural network CPU times are from the same machine (MacBook Pro, 2.2 GHz Intel Core i7 Processor, 16 GB 1600 MHz DDR3 Memory). The neural network GPU time is the running time on a Tesla P100 GPU. The same stopping criterion was used CPU and GPU neural network experiments. The superiority of the deep neural network model is clear and the computational savings would even more dramatic in 2D and 3D. Finding saddle point, local density fields with large 3D SCFT computations can take several hours whereas we should expect the proposed deep learning approach to accomplish the same task in seconds. In addition, as the size of sample set to be predicted increases, the neural network approach becomes more efficient because it is implemented based on tensor operations and the running time increases slowly as the size of input set increases.

Model	SCFT CPU	neural network CPU	neural network GPU
Prediction time	$1123s$	$5.97s$	$3.27s$

Table 3: Comparison of the direct SCFT and the deep neural network approach in terms for 200 samples.

4.3 AB_3 Star Copolymer

To show the generalizability of our model, we now test our deep learning model for an AB_3 star copolymer melt, which has a different molecular architecture than the AB diblock melt. In an AB_3 star system, three B blocks are attached at a point to the A block terminus. In both systems, strand lengths (including degeneracy factor) sum to 1. In the AB diblock case, $f_A + f_B = 1$ while $f_A + 3f_B = 1$ for the AB_3 star system.

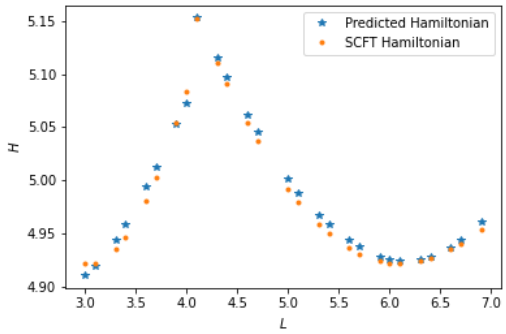
We employ the same technique for the AB_3 star system as used for the AB diblock melt, where we wrote the effective Hamiltonian as the sum of the enthalpic term (explicitly extracting the quadratic interaction) and a remainder that contains the polymer entropy and is deep learned in Sobolev space. Experiments are run on both a low-to-moderate χN case and a high χN case, and in both cases accurate results are obtained.

The predicted map from $(\chi N, L, f, \rho)$ to effective Hamiltonian H is shown in Fig. 7 and a validation of the global shift-invariance is presented in Fig. 3b. Just as for the AB diblock, the predictions for the AB_3 star copolymer are, as Table 2 quantifies, very accurate. The density profile predictions are also excellent for both low and high χN as Fig. 8 demonstrates.

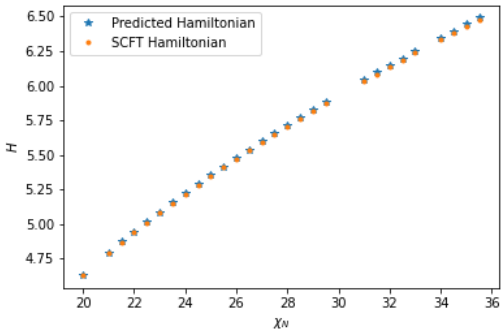
4.4 Comparison with the Kernel Ridge Regression Learner

We implemented the Sobolev-trained Kernel Ridge Regression (KRR) learner introduced in Section 3 for the AB diblock copolymer and the AB_3 star copolymer. Even though the KRR does not comply with the shift invariance constraint, a comparison with the deep NN-based method offers information on the capability of the Sobolev-trained KRR to approximate simultaneously H and its gradient, which might be useful for other applications.

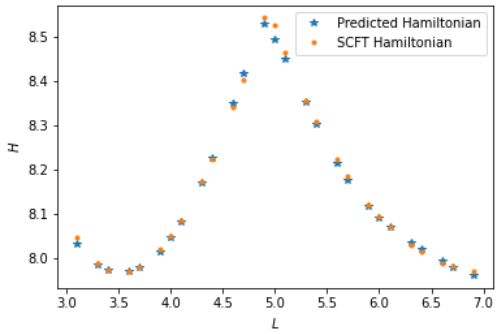
Table 4 shows a comparison of the accuracy of the deep NN and the KRR for approximating H and its gradient, for both the AB block copolymer and the AB_3 star copolymer. The accuracy in H is slightly better for the deep NN but the KRR yields a more accurate approximation of the



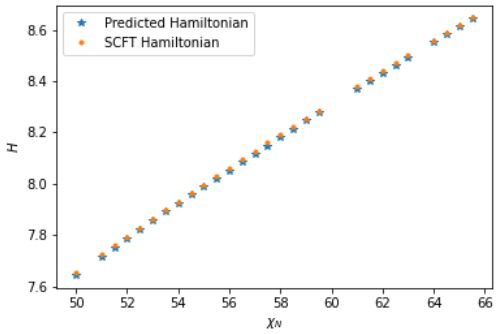
(a) $\chi N = 22$, $3 \leq L \leq 6.9$, and $f = 0.4$.



(b) $L = 5.6$, $20 \leq \chi N \leq 35.5$, and $f = 0.4$.



(c) $\chi N = 58.5$, $3 \leq L \leq 6.9$, and $f = 0.4$.



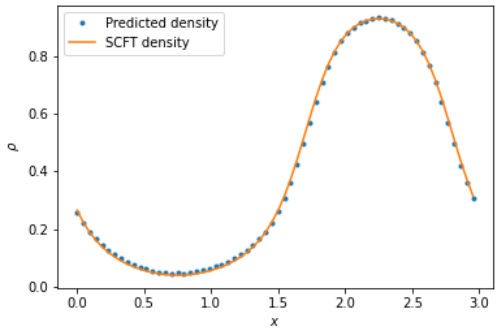
(d) $L = 5.6$, $50 \leq \chi N \leq 65.5$, and $f = 0.4$.

Figure 7: AB_3 star copolymer: comparison of predicted and SCFT effective Hamiltonian.

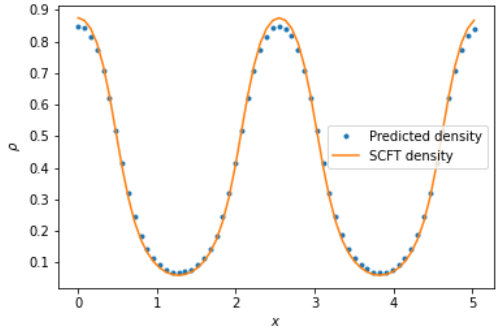
gradient. The latter is not surprising given the additional shift-invariance requirement in the deep NN which effectively reduces the approximating space. It is in fact remarkable that the deep NN yields comparable accuracy with that of the constraint-free KRR. This underlines the generalization power of the deep NN while handling additional learning constraints. On the other hand, for systems that do not require a global shift (and/or rotational) invariance, the Sobolev-trained KRR could provide a simple, accurate, and explicitly trained learner.

Structure	learner	f	E_H	E_G	E_I
AB diblock	NN	0.3	**0.017**	0.018	**0.006**
	KRR	0.3	0.021	**0.006**	–
AB_3 star	NN	0.4	**0.017**	0.065	**0.012**
	KRR	0.4	0.020	**0.007**	–

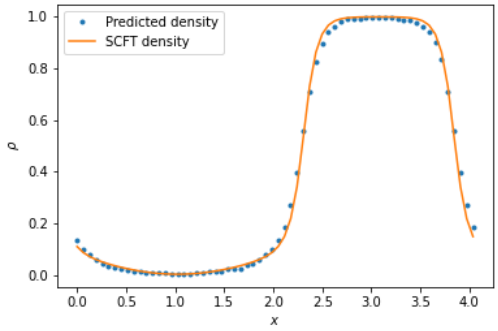
Table 4: Comparison of the NN and the KRR learners trained in Sobolev space for the AB diblock and the AB_3 star system with low χN. E_H is the error of predicted effective Hamiltonian, E_G is the error of predicted effective Hamiltonian gradient, and E_I is the difference of predicted effective Hamiltonian of shifted density fields. The KRR learner does not have this approximate shift invariance.



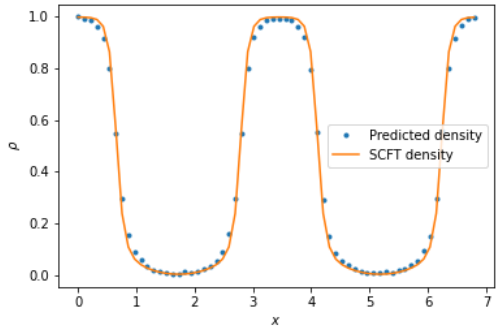
(a) $\chi N = 24.5$, $L = 3.0$, and $f = 0.4$.



(b) $\chi N = 23.0$, $L = 5.1$, and $f = 0.4$.



(c) $\chi N = 59.5$, $L = 4.1$, and $f = 0.4$.



(d) $\chi N = 64.5$, $L = 6.9$, and $f = 0.4$.

Figure 8: AB_3 star copolymer: predicted and SCFT saddle density fields.

5 Conclusions

We presented a deep learning framework to accelerate the exploration of parameter space for block copolymer systems based on field theoretic models. The central idea is to use data sets obtained from SCFT simulations to train *in Sobolev space* and learn the effective Hamiltonian of the system as a function of the relevant average monomer density field and the model parameters. The proposed neural network learner is built from rigorous universal approximation results in Sobolev space and accurately preserves global shift-invariance. Once this learning process is done, one can expeditiously find, via gradient descent, an accurate prediction for a saddle point density field. Moreover, we can potentially combine any global optimizer with this neural network approach to search for a global minimum.

The proposed SCFT-deep learning approach could also be used to accelerate the solution of the *inverse design problem*: given target properties of the system, find the parameters and composition that generate those properties. This could be done by optimizing a suitable fitness function, which measures deviations from the target system, and whose evaluation can be expeditiously done through a deep-learned functional.

There are, of course, several other machine learning models beside deep neural networks. In fact, we started this project by exploring the use of Kernel Ridge Regression (KRR), inspired by the pioneering work of Snyder, Rupp, Hansen, Blooston, Müller, and Burke [13, 14, 15] on the use of KRR in the context of density functional theory. We derived an explicit formula to compute the KRR parameters *with Sobolev training* to effectively predict H and its gradient, ∇H,

simultaneously. While this machine learning approach requires smaller training sets, we could not construct a KRR model that satisfactorily preserves the important global shift-invariance. The deep network model quickly revealed more general and robust approximations properties for our system and consequently we adopted this and not KRR.

The focus of this work has been on systems in one spatial dimension to allow us properly develop and test the proposed framework. Work on 2D and 3D systems is underway. In the higher-dimensional case we need global shift invariance along all axes, as well as rotational invariance. Both can be incorporated with the data enhancement and regularization approach proposed here.

Acknowledgments

H.D.C. and Y.X. acknowledge partial support from the National Science Foundation under award DMS-1818821. K.T.D. and G.H.F. were supported by the DMREF Program of the National Science Foundation under award DMR-1725414. Use was made of computational facilities purchased with funds from the National Science Foundation (CNS-1725797) and administered by the Center for Scientific Computing (CSC). The CSC is supported by the California NanoSystems Institute and the Materials Research Science and Engineering Center (MRSEC; NSF DMR 1720256) at UC Santa Barbara.

6 Appendix

6.1 The SCFT AB Diblock Copolymer Model

The effective Hamiltonian of an AB diblock copolymer melt is expressed in terms of two fields, w_A and w_B [1]

$$H[w_A, w_B] = \int [-(w_A + w_B)/2 + (w_B - w_A)^2/(4\chi N)] d\mathbf{r} - V \ln Q[w_A, w_B], \tag{22}$$

where χ is the Flory parameter and measures the strength of binary contacts, N is the copolymer degreee of polimerization, and V is the system volume. $Q[w_A, w_B]$ is the partition function for a single copolymer with field w_A acting on the A block and field w_B acting on the B block. This functional can be evaluated by the formula

$$Q[w_A, w_B] = \frac{1}{V} \int q(\mathbf{r}, 1; [w_A, w_B]), \tag{23}$$

where $q(\mathbf{r}, s; [w_A, w_B])$ is the copolymer propagator (s is the contour variable which parametrizes a copolymer chain) which satisfies the Fokker-Planck equation (often referred to as the modified diffusion equation)

$$\frac{\partial q}{\partial s} = \nabla^2 q - \psi q, \quad q(\mathbf{r}, 0; [w_A, w_B]) = 1. \tag{24}$$

Here,

$$\psi(\mathbf{r}, s) = \begin{cases} w_A(\mathbf{r}), & 0 \le s \le f, \\ w_B(\mathbf{r}), & f < s \le 1, \end{cases} \tag{25}$$

where f is the average volume fraction of type A blocks. The SCFT solution corresponds to saddle points of H, where H is a minimum with respect to the chemical potential-like field

$$w_-(\mathbf{r}) \equiv \frac{1}{2}[w_B(\mathbf{r}) - w_A(\mathbf{r})] \tag{26}$$

and a maximum with respect to pressure-like field

$$w_+(\mathbf{r}) \equiv \frac{1}{2}[w_B(\mathbf{r}) + w_A(\mathbf{r})]. \tag{27}$$

To find the saddle point fields, via gradient descent (for w^-) and gradient ascent (for w^+) we need to evaluate the first variation ("gradient") of H. This can be done in terms of monomer density fields ρ_A and ρ_B:

$$\begin{aligned} \frac{\delta H[w_+, w_-]}{\delta w_+(\mathbf{r})} &= \rho_A(\mathbf{r}; [w_+, w_-]) + \rho_B(\mathbf{r}; [w_+, w_-]) - 1, \\ \frac{\delta H[w_+, w_-]}{\delta w_-(\mathbf{r})} &= \frac{2}{\chi N} w_- + \rho_B(\mathbf{r}; [w_+, w_-]) - \rho_A(\mathbf{r}; [w_+, w_-]). \end{aligned} \tag{28}$$

In turn, ρ_A and ρ_B are computed using the Feynmman-Kac formulas

$$\begin{aligned}\rho_A(\mathbf{r};[w_+,w_-]) &= \frac{1}{Q[w_+,w_-]}\int_0^f q(\mathbf{r},s;[w_+,w_-])q^\dagger(\mathbf{r},1-s;[w_+,w_-])ds,\\ \rho_B(\mathbf{r};[w_+,w_-]) &= \frac{1}{Q[w_+,w_-]}\int_f^1 q(\mathbf{r},s;[w_+,w_-])q^\dagger(\mathbf{r},1-s;[w_+,w_-])ds,\end{aligned} \tag{29}$$

where the propagator $q^\dagger$ accounts for the lack of head-to-tail symmetry of the diblock. Analogously to q, $q^\dagger$ satisfies the equation,

$$\frac{\partial q^\dagger}{\partial s} = \nabla^2 q^\dagger - \psi^\dagger q^\dagger, \quad q^\dagger(\mathbf{r},0;[w_A,w_B]) = 1, \tag{30}$$

where

$$\psi^\dagger(\mathbf{r},s) = \begin{cases} w_B(\mathbf{r}), & 0 \le s < 1-f, \\ w_A(\mathbf{r}), & 1-f \le s \le 1. \end{cases} \tag{31}$$

A typical SCFT computation requires hundreds or thousands of evaluations of (28) and hence of solutions to the Fokker-Planck equations. This makes polymer SCFT simulations very expensive.

6.2 Learning H Versus Learning R Only

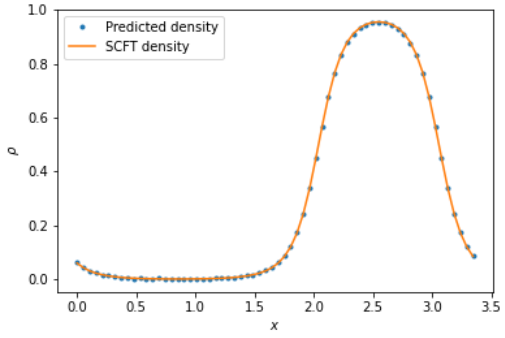
We present here numerical evidence that the proposed splitting of the Hamiltonian to focus on learning the entropic part R only produces superior results for the predicted saddle point density field than those produced by learning the full functional H. To make a fair comparison of the two strategies, we trained the two NN learners with the same architecture in the same set-up. Both methods produce comparable accuracy for H and its gradient and are both capable of enforcing shift invariance accurately but as Fig. 9 shows, the splitting approach yields significantly more accurate density field predictions.

6.3 Universal approximation

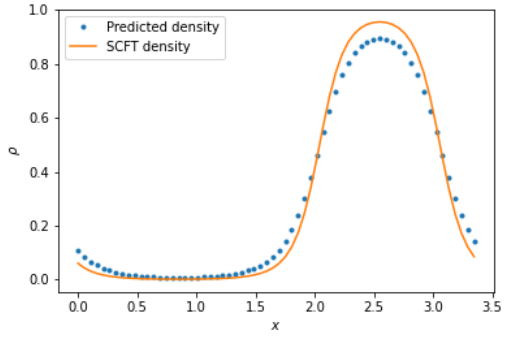
Consider the Sobolev space $W_p^m(U) \equiv \{f \in L_{1,loc}(U) | \partial^\alpha f \in L_p(U,\lambda), 0 \le |\alpha| \le m\}$. There are two significant results on the existence of neural network approximation neural network in Sobolev space. These are the following theorems.

Theorem 3. [10] If G is l-finite, $0 \le m \le l$, U is an open bounded subset of $\mathbb{R}^r$ and $C_0^\infty(\mathbb{R}^r)$ is d_p^m-dense in $W_p^m(U)$ then $\Sigma(G)$ is also d_p^m-dense in $W_p^m(U)$.

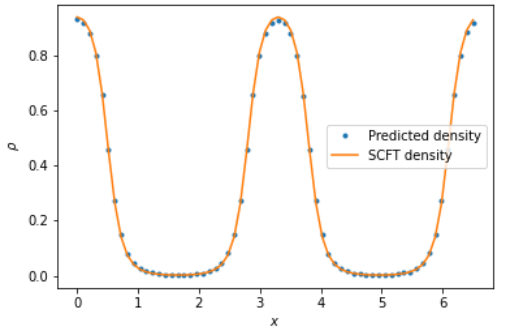
For a natural number l, G is l-finite if $G \in C^l(\mathbb{R})$ and $0 < \int |D^l G| d\lambda < \infty$. Most commonly used activation functions are l-finite. In Theorem 2 d_p^m is the Sobolev norm up to m_{th} derivative induced, by L^p norm. The condition that $C_0^\infty(\mathbb{R}^r)$ is d_p^m-dense in $W_p^m(U)$ is easy to satisfy. As pointed out by Hornik et al. [10], for all U which is a bounded domain star-shaped with respect to a point O (equivalently, any ray with origin O has a unique intersection with the boundary of U) $C_0^\infty(\mathbb{R}^r)$ is d_p^m-dense in $W_p^m(U)$. This means that for most common subsets of $\mathbb{R}^r$, the aforementioned condition is naturally true. In [10], the authors also proved several other versions of this universal approximation result for single layer neural networks with respect to the Sobolev norm. These theorems support the existence of such neural networks that simultaneously approximates a functional and its functional gradient.



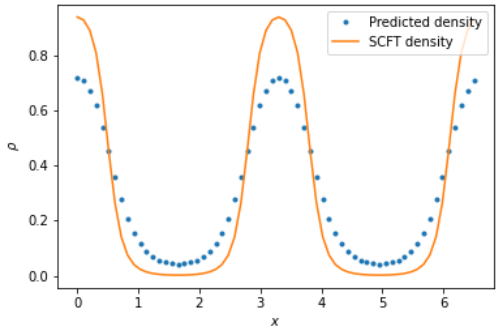
(a) $\chi N = 33.5$, $L = 3.4$, $f = 0.3$: learn the remainder



(b) $\chi N = 33.5$, $L = 3.4$, $f = 0.3$: learn the entire Hamiltonian



(c) $\chi N = 30.5$, $L = 6.6$, $f = 0.3$: learn the remainder



(d) $\chi N = 30.5$, $L = 6.6$, $f = 0.3$: learn the entire Hamiltonian

Figure 9: Learning H versus learning R only. Predicted saddle point density fields prediction for the AB diblock copolymer with low-to-moderate χN: left panels (a and c) are the predictions based on learning the entropic remainder R only, right panels (b and d) are the predictions based on learning the entire Hamiltonian H.

Theorem 4. [11] Given any two functions $f : S \to \mathbb{R}$ and $g : S \to \mathbb{R}^d$ and a finite set $U \subset S$, there exits neural network NN with a ReLU (or a leaky ReLU) activation such that $\forall x \in U : f(x) = NN(x)$ and $g(x) = \frac{\partial NN}{\partial x}(x)$.

This theorem guarantees the existence of a Sobolev space-trained neural network with 0 training loss.

6.4 Kernel Ridge Regression in Sobolev space

In this appendix, we prove Theorem 2.

Proof. Since $\tilde{H}(x) = \sum_{j=1}^{N_T} \alpha_j K(x, x_j)$, we have

$$\begin{aligned}
\nabla_\rho \tilde{H}(x) &= \sum_{j=1}^{N_T} \frac{\rho_j - \rho}{\sigma^2} \alpha_j K(x, x_j), \\
\|\nabla_\rho \tilde{H}(x_i)\|^2 &= \langle \nabla_\rho \tilde{H}(x_i), \nabla_\rho \tilde{H}(x_i) \rangle \\
&= \langle \sum_{j=1}^{N_T} \frac{\rho_j - \rho_i}{\sigma^2} \alpha_j K(x_i, x_j), \sum_{m=1}^{N_T} \frac{\rho_m - \rho_i}{\sigma^2} \alpha_m K(x_i, x_m) \rangle \\
&= \sum_{j=1}^{N_T} \sum_{m=1}^{N_T} \alpha_j \alpha_m \frac{1}{\sigma^4} K(x_i, x_j) K(x_i, x_m) \langle \rho_j - \rho_i, \rho_m - \rho_i \rangle \\
&= \alpha^T \tilde{K}_i \alpha,
\end{aligned}$$

where $\tilde{K}_i$ is a matrix with $(\tilde{K}_i)_{jm} = \frac{1}{\sigma^4} K(x_i, x_j) K(x_i, x_m) \langle \rho_j - \rho_i, \rho_m - \rho_i \rangle$. Thus,

$$\sum_{i=1}^{N_T} \|\nabla_\rho \tilde{H}(x_i)\|^2 = \sum_{i=1}^{N_T} \alpha^T \tilde{K}_i \alpha = \alpha^T \tilde{K} \alpha, \tag{32}$$

where $\tilde{K} = \sum_{i=1}^{N_T} \tilde{K}_i$. Note that both $\tilde{K}_i$ and $\tilde{K}_i$ are symmetric matrices. The cost function (16) can now be rewritten as

$$\begin{aligned}
C(\alpha) &= \sum_{i=1}^{N_T} (\tilde{H}(x_i) - H_i)^2 + \beta\, \alpha^T \tilde{K} \alpha + \lambda\, \alpha^T \tilde{K} \alpha \\
&= (H - K\alpha)^T (H - K\alpha) + \beta\, \alpha^T \tilde{K} \alpha + \lambda\, \alpha^T K \alpha,
\end{aligned} \tag{33}$$

Therefore

$$\frac{\partial C}{\partial \alpha} = 2(-K^T)(H - K\alpha) + \beta(\tilde{K} + \tilde{K}^T)\alpha + \lambda(K + K^T)\alpha = 0 \tag{34}$$

and consequently, α is the solution of the linear system

$$(K^T K + \beta \tilde{K} + \lambda K)\alpha = K^T H. \tag{35}$$

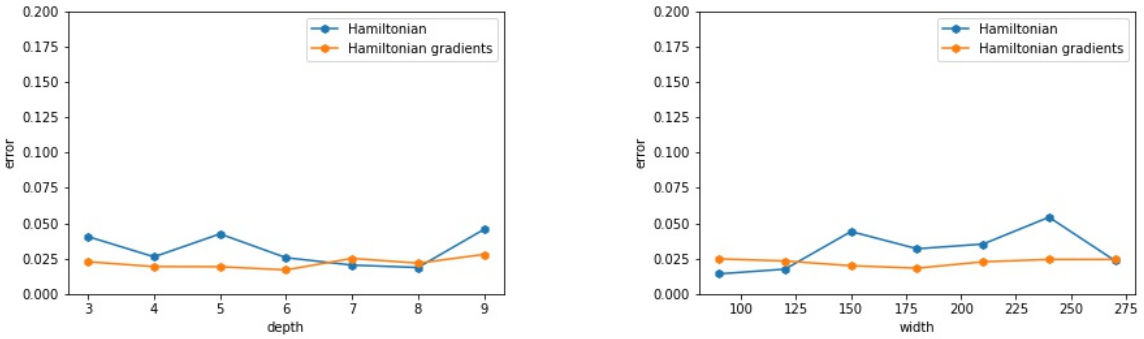
□

6.5 Ablation Study

In this section, we summarize results of a study that guided our choice of the network size. We consider the approximation error, in both H and its gradient, as the networks depth (number of hidden layers) and the network width (number of cells per layer) is changed while the training and test sets and all the other hyperparameters are kept fixed. We also examine the behavior of the training and validation loss functions as the number of epochs (iterations on the stochastic gradient descent method) increases.

In the first batch of experiments, all the hyperparameters are fixed except for the network depth. Seven neural networks with different number of hidden layers are trained on the same training set and evaluated on the same test set independently. In the second batch of experiments,

the number of cells (network width) in the middle hidden layers are the only variable while all other hyperparameters are fixed. Again, seven neural networks with different widths are trained on *the same training set* and evaluated on the same test set. Figure 10 shows that the approximation is relatively stable with respect to the network size (both in depth and width) for the range considered. This suggests that for the relatively modest training set size there is no discernible improvement of the approximation as the network size grows beyond 6-8 levels. In other words, the training set is not big enough to benefit from the use of larger networks. This observation is consistent with the results reported by D'souza, Huang, and Yeh [20] on a convolutional neural network with a small sample size.



(a) error vs. number of hidden layers

(b) error vs. number of cells in middle hidden layers

Figure 10: Relationship between the error on test set (root square error of H and its gradient) and the size of neural network: (a) error versus number of hidden layers (depth), (b) error versus number of hidden cells (width) from hidden layer 2 to hidden layer 5.

We now look at the behavior of the training and validation loss functions as number of epochs (iterations on the stochastic gradient descent method) increases. The training loss functions and the validation loss function (the sum of the loss of the effective Hamiltonian, the effective Hamiltonian gradient and the shift invariance term) after 100 epochs are presented in Fig. 11. Both training loss

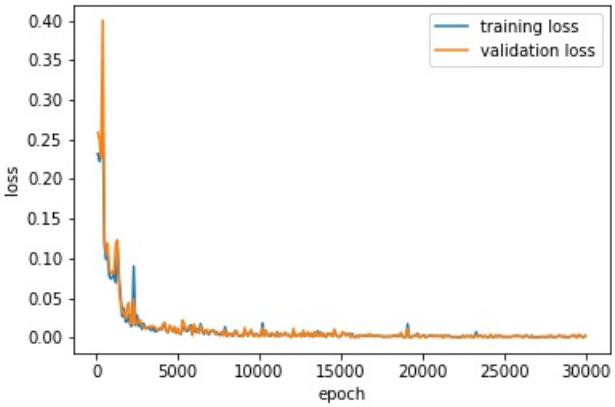


Figure 11: Training and validation loss functions against the number of epochs (stochastic gradient descent iterations).

function and the validation loss function (not used in training) decrease as expected as the number of epochs increases.

6.6 Hyperparameters Employed in the Numerical Experiments

We list in Table 5 the hyperparameters we used for the experiments in Section 4. We took $\beta = 1$

χN	Structure	f	learning rate	λ	β	γ
low χN	AB diblock	0.3	0.0001	8.15e-08	1	$N_s/(N_s-1)$
		0.4	0.001	8.44e-09	1	$N_s/(N_s-1)$
	AB_3 star	0.4	0.0001	5.23e-08	1	$N_s/(N_s-1)$
high χN	AB diblock	0.3	0.001	4.97e-10	1	$N_s/(N_s-1)$
		0.4	0.001	2.80e-10	1	$N_s/(N_s-1)$
	AB_3 star	0.4	0.001	4.93e-11	1	$N_s/(N_s-1)$

Table 5: Hyperparameters employed in the numerical experiments. N_s is the number of possible shifts.

to stress the equal priority of approximating both the Hamiltonian and its gradient also to match the coefficient of the Sobolev norm. We chose $\gamma = N_s/(N_s - 1)$, which is close to 1. This choice takes into account that one of the possible shifts, $s = N_s$, is a one-period shift and leads to a 0 difference with $NN(x_i)$ in formula (10) (only $N_s - 1$ valid terms in the summation). We employed the hyperparameters-tuning strategies described in Section 2.4 to select the optimal learning rate and the regularization coefficient λ. We took $\beta_V = \beta$ and $\gamma_V = \gamma$ in the validation loss (13) and $\theta_V = N_s$ to balance the summation. Note that one could also view β and γ as hyperparameters and tune them as done for the learning rate and λ, which might generate better results. We did not do this because with the more expedited use of fixed values of β and γ the deep NN already produced impressive results.

In Step 2, when we search for saddle density fields by gradient descent, we performed 500 iterations after selecting the initial density fields from training set.

References

[1] G.H. Fredrickson. *The equilibrium theory of inhomogeneous polymers*. Oxford University Press, 2006.

[2] M. W. Matsen. *Self-Consistent Field Theory and Its Applications*, chapter 2, pages 87–178. John Wiley & Sons, Ltd, 2007.

[3] F. Schmid. Self-consistent-field theories for complex fluids. *Journal of Physics: Condensed Matter*, 10(37):8105–8138, sep 1998.

[4] H. D. Ceniceros and G. H. Fredrickson. Numerical solution of polymer self-consistent field theory. *Multiscale Modeling & Simulation*, 2(3):452–474, 2004.

[5] P. Stasiak and M. W. Matsen. Efficiency of pseudo-spectral algorithms with Anderson mixing for the SCFT of periodic block-copolymer phases. *The European Physical Journal E*, 34(10):1–9, 2011.

[6] Issei Nakamura. Phase diagrams of polymer-containing liquid mixtures with a theory-embedded neural network. *New Journal of Physics*, 22(1):015001, 2020.

[7] Qianshi Wei, Ying Jiang, and Jeff ZY Chen. Machine-learning solver for modified diffusion equations. *Physical Review E*, 98(5):053304, 2018.

[8] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[9] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.

[10] K. Hornik, M. Stinchcombe, and H. White. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural networks*, 3(5):551–560, 1990.

[11] W. M. Czarnecki, S. Osindero, M. Jaderberg, G. Swirszcz, and R. Pascanu. Sobolev training for neural networks. In *Advances in Neural Information Processing Systems*, pages 4278–4287, 2017.

[12] R. Zhang. Making convolutional networks shift-invariant again. *CoRR*, abs/1904.11486, 2019.

[13] J. C. Snyder, M. Rupp, K. Hansen, K. R. Müller, and K. Burke. Finding density functionals with machine learning. *Physical review letters*, 108(25):253002, 2012.

[14] J. C. Snyder, M. Rupp, K. Hansen, L. Blooston, K. R. Müller, and K. Burke. Orbital-free bond breaking via machine learning. *The Journal of chemical physics*, 139(22):224104, 2013.

[15] J. C. Snyder, M. Rupp, K. R. Müller, and K. Burke. Nonlinear gradient denoising: Finding accurate extrema from inaccurate functional derivatives. *International Journal of Quantum Chemistry*, 115(16):1102–1114, 2015.

[16] K. Rasmussen and G. Kalosakas. Improved numerical algorithm for exploring block copolymer mesophases. *Journal of Polymer Science Part B: Polymer Physics*, 40(16):1777–1783, 2002.

[17] G. Tzeremes, K. Rasmussen, T. Lookman, and A. Saxena. Efficient computation of the structural phase behavior of block copolymers. *Physical Review E*, 65(4):041806, 2002.

[18] M. Chen, H. Jiang, W. Liao, and T. Zhao. Nonparametric regression on low-dimensional manifolds using deep relu networks. *arXiv preprint arXiv:1908.01842*, 2019.

[19] J. Lu, Z. Shen, H. Yang, and S. Zhang. Deep network approximation for smooth functions. *arXiv preprint arXiv:2001.03040*, 2020.

[20] N. D'souza R, P.-Y. Huang, and F.-C. Yeh. Structural analysis and optimization of convolutional neural networks with a small sample size. *Scientific Reports, Nature*, 10(1):834, 2020.

[21] H. D. Ceniceros. Efficient order-adaptive methods for polymer self-consistent field theory. *Journal of Computational Physics*, 386:9–21, 2019.