Structural Prediction and Inverse Design by a Strongly Correlated Neural Network

Jianfeng Li* and Hongdong Zhang
The State Key Laboratory of Molecular Engineering of Polymers, Department of Macromolecular Science, Fudan University, Shanghai 200433, China

Jeff Z. Y. Chen†
Department of Physics and Astronomy, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1

(Received 14 February 2019; revised manuscript received 1 June 2019; published 4 September 2019)

Macromolecules contain molecular units as the coding information for their correlated structures in physical dimensions. The relationship between these two features is governed by the interaction energies of the involved molecular units and their encoded sequences. We present a neural network algorithm that treats molecular units themselves as neural networks, which has the flexibility to allow each unit to respond to its own environment and to influence others in the system. Through a deep neural network and a self-consistent procedure, molecular units in the network establish a strong correlation to produce the desirable features in the physical world. The proposed framework is applied to the *HP* model. Both the forward problem of predicting folded structures from given sequences and the inverse problem of predicting required sequences for a given structure are examined.

DOI: 10.1103/PhysRevLett.123.108002

Introduction.—Many structural-prediction problems in polymer physics and biological physics start with the primary coding of a variety of basic molecular units at a molecular level. In block copolymers, the coding could be blocks of monomers connected together where each block is of a particular entity; in protein folding, the coding could be the arrangement of amino acids along a linear backbone. These are either artificially synthesized or naturally made linear "texts" that have, at the first glance, no information on their three-dimensional structures. Through interactive potential energies between the basic molecular units (with the possible solvent involvement) and under appropriate physical conditions, these linear or linearly branched chains self-assemble (or "fold") into well-defined structures in physical space.

Theoretically, going from a sequence text to its three-dimensional structure, one employs procedures established according to the basic principles of statistical mechanics, taking, for example, Monte Carlo simulations [1], molecular dynamics simulations [2], or self-consistent field theory calculations [3]. Experimentally, one just measures the resultant structures with the understanding that some physical processes have taken place in between the sequence information and the final structures, without knowing the abstract concepts such as the Hamiltonian or the Boltzmann distribution of the system. Computationally, if all the physical processes can be embedded in a well-constructed neutral network (NN), one directly establishes a relationship between the final structure and the starting sequence text, letting the network to handel everything in the middle. The strongly correlated neural-network (SCN) introduced here is designed for such a purpose, which treats physical feature 1 as input (e.g., sequence information) and feature 2 as output (e.g., spatial arrangement of molecules). It can also be used for inverse design by assigning the resultant structures as feature 1 and asking for the original sequence texts as feature 2 (see Fig. 1).

Using NNs as a computational tool to perform dimensional reduction for use in condensed matter is now common [4,5]. When position coordinates in an off-lattice model or the location and value of spins in an Ising model are given, the multidimensional information can be reduced to a simple few, for identification of the main physical characteristics such as order parameters of different phases, with the implementation of relatively elementary NN models [6–10]. The use of NNs has also inspired unconventional ways of calculating the partition function [11–13], conducting Monte Carlo simulations [14–17], and solving partial differential equations [18–20].

Instead of dimension reduction, our task here is to establish the ability to describe both features 1 and 2 in much molecular detail. The SCN contains two basic elements. The first is to associate the molecular-level information in feature 1 (the input) with an identity layer of NNs, shown in Fig. 1. This expanded concept of assigning an NN to a physical entity allows for the maximal flexibility of such an entity to react in response to a complex physical environment. The response from each identity NN is then sent to a convolutional neural network (CNN) for further crossing analysis. The second is a self-consistent loop that collects the information from the CNN and feeds back to the identity layer. This establish a strong

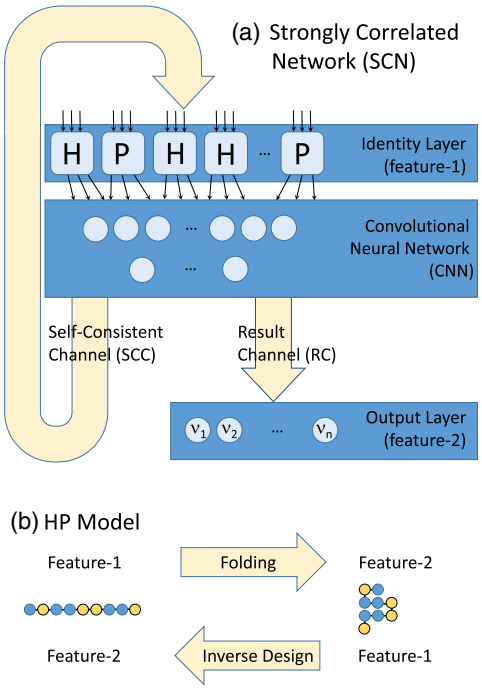

FIG. 1. (a) Sketch of the strongly correlated network (SCN) and (b) the physical example used for demonstration (*HP* model). SCN represents entities in physical feature 1 by neural networks and sends their output to a deep neural network for processing. Two output channels are designed: one for feeding back to the input of the identity layer and the other for the calculated physical feature 2. In the folding example of the *HP* model, feature 1 is the encoding sequence and feature 2 is the coordinates of the folded structure. In inverse design, feature 1 is the desired fold and feature 2 is the predicted sequence that in turn produces the fold. The network mimics the physical process that connects features 1 and 2.

correlation between feature 1 in the information space by itself. After the self-consistency is achieved, feature 2 is then extracted.

Any NN applications of a physical problem deal with two categories of data. The physical data is used to represent the physical properties of a particular problem, not different from the measurements taken from a real laboratory. The network parameters are used to build NNs through function calls, normally invisible to the users. Once the feature 2 output data is fitted to a real physical data by adjusting network parameters, then the SCN is thought of as trained.

In order to show the power of SCN, we take a well-known model in statistical physics, the *HP* model [21], as the vehicle to deliver the concepts. The sequence of a theoretical protein is encoded in a linear chain with two types of molecular units, *H* for hydrophobic and *P* for hydrophilic [Fig. 1(b)]. Some proteinlike sequences fold into unique native structures. The folding problem asks for prediction of the native structure (feature 2), when a sequence (feature 1) is given. The inverse design problem can also be formulated: can we predict the encoding sequences (feature 2) that fold into a given native structure (feature 1)? The 19mer *HP* model used here is complicated enough to contain sufficiently big data, and yet is simple enough to have all structural properties exactly determined previously.

SCN.—Our network design is conceptually shown in Fig. 1, containing the following main components. The first is an identity layer. For example, molecular units are connected sequentially in the *HP* model to form a one-dimensional text and there are only two physical entities, *H* and *P*. Each entity is represented by a feed-forward-neural-network type. From one *H* to another *H*, the network parameters are identical to represent the same entity, but the input and output vectors can be different. Hence, each *H* network individually takes its unique environment through the input and responds to it by producing its own behavior through the output. One could view the network representation of a physical entity as the numerical realization of a mathematical operator; by reacting to the various inputs, it projects its own properties onto different outputs. All *P* networks have the same properties. The concept is vastly different from previous representations of these molecular units by fixed numbers in one hot encoding [22] (for example, 0 for *H* and 1 for *P*) or by vectors with limited adaptability [23–25].

To enable a mechanism for strong correlations between units in feature 1, the outputs from the identity layer are directly sent to the input of a CNN, where all these outputs are physically correlated through nonlinear numerical mixing. The CNN output is divided into two channels, a self-consistent channel (SCC) and a result channel (RC). The SCC is a vector output, with the same vector dimension as the total dimension of the input to the identity layer. During the calculation, the self-consistency is ensured by looping the information from the SCC back to the input of the identity layer until self-convergence. Making a mathematical comparison, we can denote the input to the identity layer by a vector $\mathbf{v}$ and the combined effects from identity layer and CNN, sent to the SCC, by a vector function $\mathbf{F}(\mathbf{v})$; then the self-consistent loop seeks the convergence to a fixed-point solution $\mathbf{v} = \mathbf{F}(\mathbf{v})$. Once this is established, the molecular units are strongly correlated to each other in a large network space.

The convergence, of course, depends on the network parameters in both identity layer and CNN. The RC gives a set of property data of the SCN after the self-consistent loops. In supervised training, this set of data is then forced to follow the known physical data of feature 2; through minimizing a cost function, the network parameters are adjusted to yield the optimal performance. The SCN is then considered trained and ready for making predictions.

Structural prediction in HP model.—To demonstrate the usage of SCN, we consider the folding of 19mer *HP* chains into two-dimensional (2D) structures on a square lattice

[3,26–28]. Feature 1 is the serial encoding of 19 monomers where each monomer can be an H or P. A monomer is allowed to occupy an unoccupied lattice site and double occupancy is disallowed. An H-H nearest-neighbor contact reduces the total free energy of the system by ϵ and no other interactions are assumed. This effectively pushes the H units inside a cluster. A properly designed sequence of HP protein has a uniquely folded 2D native structure. This rather essential model is selected here because all possible sequences have been enumerated and the exact solutions are known [29–31]; in this way we can benchmark SCN's performance. In total 13 454 different sequences were found to have unique native states, whose folded coordinates are exactly known, forming our feature 2 [32]. These 13 454 HP sequences and their corresponding native states are randomly divided into two groups: 11 454 samples were used for training and the remaining 2000 samples form an independent dataset to test the accuracy of the trained SCN.

Here are a few technical details. The identity layer contains $L = 19$ feed-forward networks of two entities, H and P, each having one hidden layer of 50 neurons. These NNs, $l = 1, 2, 3, \ldots, L$, represent the L monomers along the backbone of a specifically sequenced chain. The input into and output from each identity network are eight dimensional vectors. The $L \times 8$ output nodes are used as the input to the CNN, which contains 2 convolutional layers and several fully connected layers. The output from CNN are two channels, SCC and RC. The SCC also contains $L \times 8$ output nodes to be directly connected to the input of the identity layer. In practice, from a random initialization, 10 self-consistency loops are taken before the RC is used. The RC is an $(L-2) \times 3$ matrix to specify the coordinates of all monomers starting from $l = 3$. The group of 3 numbers $\nu_{l,j}$, $j = 1, 2, 3$, respectively, yield the probabilities of the three folding directions (left, forward, or right) on a 2D lattice that the lth monomer takes from the $(l-1)$th, and depend on the network parameters. In supervised training, the cross-entropy cost function is minimized in multiple steps with respect to all network parameters in the identity layer and CNN,

$$S = -\sum \nu^*_{l,j} \ln \nu_{l,j}, \tag{1}$$

where $\nu^*_{l,j}$ is the target value specified by the samples taken from the training dataset. An epoch is a computational step when the entire training dataset is used once, on average [22].

As the network is trained epoch by epoch, we monitor the SCN performance by feeding the sequence data from the independent test dataset and comparing the coordinate outputs with the known results. An accuracy A is defined as the percentage of the 2000 molecular-coordinate datasets that are *exactly* reproduced. The black curve in Fig. 2 indicates that, by the end of supervised training, nearly 80% of the native structures in the test dataset can be predicted by SCN.

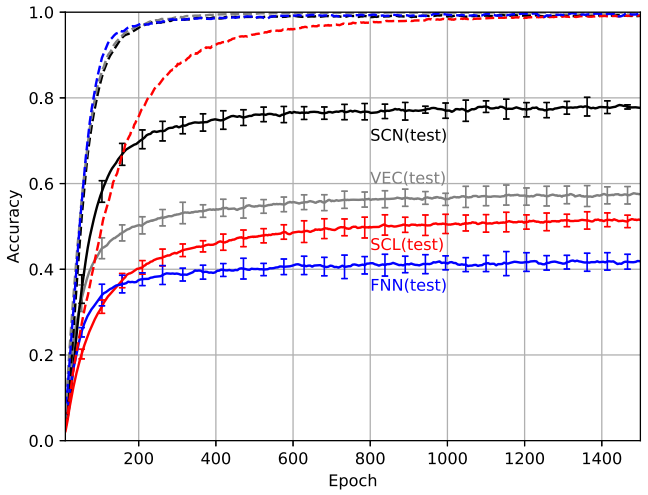

FIG. 2. Accuracy of SCN to produce the correct native structures in the independent test data of the 19mer HP model, as a function of the supervised-training epoch (solid black curve). Other deep NNs, such as vector (VEC), scalar (SCL), and fully connected (FNN) models have prediction accuracies represented by the solid gray, red, and blue curves. The efficiencies of the supervised training are monitored by the dashed black, gray, red, and blue curves, representing the accuracies measured on the training dataset itself, for SCN, VEC, SCL and FNN, respectively. [33].

The efficiencies of using other deep NNs of a similar network size are compared here. The state-of-the-art folding NN uses vector (VEC) representation in the identity layer [36,37], instead of the network representation proposed in Fig. 1. A typical entity (e.g., H) is represented by a vector, whose (e.g., 8) elements are free network parameters determined by training. In a scalar model (SCL), such an identity is represented by a static scalar (e.g., 0 for H and 1 for P); a SCL is essentially a CNN. A fully connected deep NN (FNN) is also examined. No feeding back loop to reinforce the environment-identity correlation is designed in the conventional VEC, SCL, and CNN [33]. We used the same HP model and partitioning of the training and testing datasets to train these models. The gray, red, and blue curves in Fig. 2 show that these deep NNs underperform SCN by a significant margin, measured by A. The only drawback of SCN is that depending on the self-consistency loops that it takes, the real computational time can be a few times slower than VEC; this can be tolerated in view of the much better accuracy.

One-to-many prediction and inverse design.—Shown in Fig. 1, the SCC feeds to the identity layer with the collective environment information. When a calculation starts, no such information exits yet and a random environment parametrization is assumed; as the self-consistent loop proceeds, it is assumed to converge to a fixed point. This opens the door for exploiting another important

property of SCN: under the same set of appropriately-selected network parameters and depending on the initial random environment parameters, SCN has a built-in mechanism to converge to multiple fixed points through the self-consistent loops; that is, with the same physical feature 1, it could make multiple predictions on feature 2, which is the ability that VEC lacks.

This is particularly useful in solving the inverse-design problem. Given originally desired feature 2, what prediction can we make to design feature 1 in order to make feature 2 happening. Now reversing the roles of features 1 and 2, the SCN can be readily used for supervised training to answer this question, however, with one caveat. There could be many selections of the original feature 1 that give rise to the same feature 2. For example, in the 19mer *HP* model, it is well documented that on average $g \approx 10$ sequences, sometimes as many as $g = 64$, fold into a single, uniquely identifiable native structure, where g is the degrees of degeneracy. Once the roles are reversed, the predictive network must have the ability to predict different sequences (now feature 2), given the same native structure (now feature 1).

Technically, the identity layer contains a chain of $(L-2)$ NNs of three entities, representing the coordinates of a native structure. The three entities are simply "left," "forward," and "right," indicating how the lth monomer turns in a 2D space from the $l-1$ monomer by taking a configurational step. For feature 2, the output from CNN is a $L \times 2$ matrix $v_{l,j}$ where the two output nodes for each l predict the probability that the lth monomer is H $(j = 0)$ or P $(j = 1)$. The same cross entropy, Eq. (1), is used in the supervising training by taking the training dataset. There are 1345 folded structures in the 19mer dataset, which are divided into two different sets. The first 1145 folded structures (from 12 482 sequences) are treated as the training dataset and the second 200 folded structures (from 972 sequences) the test dataset.

Once the SCN learns through the training dataset by supervised training, the network is ready for making inverse predictions. To produce a sequence for a specifically desirable structure, we initialize the SCN by M sets of random environmental inputs (i.e., M prediction events) to encourage the network to converge to different fixed points (hence to produce the degenerated sequences). After all folded structures in the test dataset are considered, two measurements are taken to benchmark the success of the prediction. Designability D is the percentage of the folded states in the test set that have at least one sequence accurately predicted. Recoverability R is the ratio between the number of all correctly predicted sequences and the total numbers of different sequences (i.e., 972) on the test set. Of course, both D and R are functions of M, as multiple prediction events are needed to produce different sequences [see Fig. 3(a)].

Among the test dataset of the folded structures, we can further categorize them by the degrees of degeneracy g. The rotated labels in Fig. 3(b) indicate the fraction of the folded structures in the dataset that have a particular g. In general, as more prediction events are taken (M progresses), more known structures are recovered at a given g.

Summary.—This work proposes to represent the classical version of a physical entity by a neural network for use in artificial intelligence. This accommodates a computational mechanism to embed physical interactions with other molecular units in a neural-network form, influenced by other molecules through input and responding through output. Incorporating a further deep neural network, we also propose to establish a strongly correlated network system by a self-consistent procedure, to approximate the entire physical system where the specified molecular units correlate themselves to form definitive structures in physical space. We establish a network framework that has the ability to capture the strong structural correlations at the

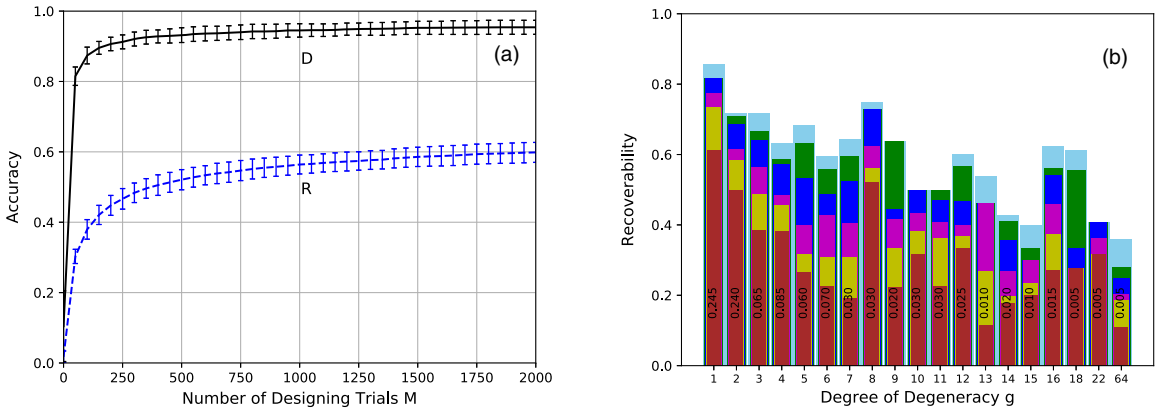

FIG. 3. (a) Designability D and recoverability R of the trained SCN in making inverse-design predictions on the independent test dataset and (b) recoverability R further categorized by the degrees of degeneracy g [33]. These measures are plotted as functions of M, the number of designing trials made on the trained SCN. In plot (b), the six colors on the bar graph represent $M = 50$, 100, 200, 500, 1000, and 2000, from bottom to top, respectively. The rotated labels associated with vertical bars in (b) indicate the fraction of structures in the testing dataset that have the corresponding g.

molecular level, resulted from laws of physics which are unseen by the network, through setting up network parameters by supervised learning.

We demonstrate the usage of the network by supervising it to solve two physical problems related to the folding of "theoretical proteins" known as the *HP* model. In the forward problem, the network learns how to predict the structures of a folded protein in physical space from given sequence information, with no additional interaction potential energies specified. In the inverse problem, the network is asked to prescribe a set of sequence information, on the basis of the molecular coordinates of a folded structure without knowing the folding kinetics.

Depending on the physical system to be modeled, the self-consistent loop can converge to multiple fixed-point solutions. This allows for many-to-one, one-to-many (both demonstrated here), and many-to-many predictions (not demonstrated) at the molecular level, making solving degenerate states possible [38,39].

We thank the financial support from the National Natural Science Foundation of China (Grants No. 21534002, No. 21973018, and No. 21873009) and the Natural Sciences and Engineering Research Council (NSERC) of Canada. J. Z. Y. C. thanks the University of Waterloo for a seeding International Collaboration Grant. The computation was made possible by the facilities of Compute Canada.

*lijf@fudan.edu.cn
†jeffchen@uwaterloo.ca

[1] E. Shahknovich, G. Farztdinov, A. M. Gutin, and M. Karplus, Phys. Rev. Lett. **67**, 1665 (1991).
[2] H. A. Scheraga, M. Khalili, and A. Liwo, Annu. Rev. Phys. Chem. **58**, 57 (2007).
[3] H. Duan, J. F. Li, H. D. Zhang, and F. Qiu, Polymer **134**, 75 (2018).
[4] L. Zdeborová, Nat. Phys. **13**, 420 (2017).
[5] P. Mehta, M. Bukov, C. K. Fisher, and D. J. Schwab, Phys. Rep. **810**, 1 (2019).
[6] J. Carrasquilla and R. G. Melko, Nat. Phys. **13**, 431 (2017).
[7] Q. Wei, R. G. Melko, and J. Z. Y. Chen, Phys. Rev. E **95**, 032504 (2017).
[8] S. J. Wetzel and M. Scherzer, Phys. Rev. B **96**, 184410 (2017).
[9] A. Morningstar and R. G. Melko, J. Mach. Learn. Res. **18**, 1 (2018).
[10] M. J. S. Beach, A. Golubeva, and R. G. Melko, Phys. Rev. B **97**, 045207 (2018).
[11] G. Torlai and R. G. Melko, Phys. Rev. B **94**, 165134 (2016).
[12] C. Desgranges and J. Delhommelle, J. Chem. Phys. **149**, 044118 (2018).
[13] D. Wu, L. Wang, and P. Zhang, Phys. Rev. Lett. **122**, 080602 (2019).
[14] L. Huang and L. Wang, Phys. Rev. B **95**, 035105 (2017).
[15] L. Huang, Y.-f. Yang, and L. Wang, Phys. Rev. E **95**, 031301 (2017).
[16] L. Wang, Phys. Rev. E **96**, 051301 (2017).
[17] W. Yu, Y. Liu, Y. Chen, Y. Jiang, and J. Z. Y. Chen, J. Chem. Phys. **151**, 031101 (2019).
[18] J. Han, A. Jentzen, and W. E, arXiv:1707.02568.
[19] W. E, J. Han, and A. Jentzen, Commun. Math. Stat. **5**, 349 (2017).
[20] Q. Wei, Y. Jiang, and J. Z. Y. Chen, Phys. Rev. E **98**, 053304 (2018).
[21] H. S. Chan and K. A. Dill, J. Chem. Phys. **100**, 9238 (1994).
[22] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, Cambridge, MA, 2016).
[23] T. Mikolov, K. Chen, G. Corrado, and J. Dean, arXiv: 1301.3781v3.
[24] T. Mikolov, I. Sutskever, G. Corrado, and J. Dean, arXiv: 1310.4546v1.
[25] X. X. Chen, Z. Y. Liu, and M. S. Sun, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (Association for Computational Linguistics, Doha, 2014), p. 1025.
[26] K. A. Dill, Biochemistry **24**, 1501 (1985).
[27] K. F. Lau and K. A. Dill, Macromolecules **22**, 3986 (1989).
[28] K. A. Dill, Biochemistry **29**, 7133 (1990).
[29] A. Sali, E. Shahknovich, and M. Karplus, Nature (London) **369**, 248 (1994).
[30] R. Unger and J. Moult, J. Mol. Biol. **231**, 75 (1993).
[31] K. Yue and K. A. Dill, Proc. Natl. Acad. Sci. U.S.A. **92**, 146 (1995).
[32] Please see https://github.com/vvoelz/HPSandbox.
[33] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevLett.123.108002 for a detailed description of SCN, VEC, SCL, and FNN used here, and the method used for calculating the error bars, which refers to Refs. [34,35].
[34] Please see https://www.tensorflow.com.
[35] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Proc. IEEE **86**, 2278 (1998).
[36] S. Wang, J. Peng, J. Z. Ma, and J. B. Xu, Sci. Rep. **6**, 18962 (2016).
[37] R. Evans, J. Jumper *et al.*, De novo structure prediction with deep-learning based scoring, in *Thirteenth Critical Assessment of Techniques for Protein Structure Prediction (Abstracts)* (Protein Structure Prediction Center, Riviera Maya, 2018).
[38] D. J. Liu, Y. X. Tan, E. Khoram, and Z. F. Yu, ACS Photonics **5**, 1365 (2018).
[39] G. White and W. Seffens, Electron. J. Biotechnol. **1**, 196 (1998).