Editors' Suggestion   Featured in Physics

Identifying knot types of polymer conformations by machine learning

Olafs Vandans,[1] Kaiyuan Yang,[2] Zhongtao Wu,[3] and Liang Dai[1,4,*]

[1]*Department of Physics, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon, Hong Kong, China*
[2]*Department of Computer Science, School of Computing, National University of Singapore, Singapore 117417, Singapore*
[3]*Department of Mathematics, The Chinese University of Hong Kong, Shatin, NT, Hong Kong, China*
[4]*Shenzhen Research Institute, City University of Hong Kong, Shenzhen, China*

(Received 14 November 2019; accepted 14 January 2020; published 11 February 2020)

We investigate the use of artificial neural networks (NNs) as an alternative tool to current analytical methods for recognizing knots in a given polymer conformation. The motivation is twofold. First, it is of interest to examine whether NNs are effective at learning the global and sequential properties that uniquely define a knot. Second, knot classification is an important and unsolved problem in mathematical and physical sciences, and NNs may provide insights into this problem. Motivated by these points, we generate millions of polymer conformations for five knot types: 0, 3_1, 4_1, 5_1, and 5_2, and we design various NN models for classification. Our best model achieves a five-class classification accuracy of above 99% on a polymer of 100 monomers. We find that the sequential modeling ability of recurrent NNs is crucial for this result, as it outperforms feed-forward NNs and successfully generalizes to differently sized conformations as well. We present our methods and suggest that deep learning may be used in specific applications of knot detection where some error is permissible. Hopefully, with further development, NNs can offer an alternative computational method for knot identification and facilitate knot research in mathematical and physical sciences.

DOI: 10.1103/PhysRevE.101.022502

I. INTRODUCTION

Knotting is a common phenomenon occurring in everyday objects and polymers [1,2], including DNA [3,4] and proteins [5,6]. The prevalence of knots has attracted the interest of scientists from a broad range of disciplines, not only mathematicians but also polymer physicists studying the effect of knotting on polymer conformations and dynamics [7–10]. In addition, molecular biologists are interested in the roles of knotting in the biological functions of protein knots [11,12] and DNA knots [13,14], and chemists are motivated to synthesize knotted molecules [15] and investigate their special catalysis [16]. From the viewpoint of materials, knotted structures can be designed and used for practical applications [17–21]. Furthermore, recently technologists have encountered DNA knots in nanopore sequencing [22–24] and nanochannel-based genome mapping [25–27].

Addressing a wide range of problems about knots is based on one foundation: precise and efficient identification of the knot type for a given conformation. The *knot recognition or classification problem* is the most fundamental problem in knot theory, which asks whether two given knots are equivalent. Like many other problems in the theory of computation, there are two aspects of the questions that one wants to answer: (i) Is the problem algorithmically decidable? (ii) If so, what is the complexity? Question (i) was first answered in the affirmative by Haken [28] using his theory of a normal surface [29]. Unfortunately, the algorithm has an exponential running time; and indeed, no efficient algorithm is known for question (ii). Even for the special case of the *unknot recognition problem*, it is unknown whether a polynomial time algorithm exists.

The classical approach to knot classification is to develop invariants that are shared by equivalent knots. Some of the well-known invariants include (i) the Alexander, Jones, and other knot polynomial invariants [30–32]; (ii) the Heegaard-Floer, Khovanov, and other knot homology invariants [33,34]. Typically, there is a tradeoff between complexity and the ability to recognize knots. For instance, while it takes polynomial time to compute the Alexander polynomial, there are infinitely many inequivalent knots that share the same Alexander polynomial of the unknot. On the other hand, although both Heegaard-Floer and Khovanov homology can recognize the unknot [35], the computation time of those homologies is extremely large.

In this work, we attempt to determine knot types for given conformations based on machine learning, which is different from the above-described methods based on analytic theory. Machine learning models are *trained* via a data-processing task, and the *learning* takes place by adjusting to the feedback signal during training [36,37]. Deep learning is a specific subfield of machine learning, with an emphasis on successive layers of data representations [38–40]. In deep learning, these layered data-encoding models are referred to as *neural networks* (NNs). Recently, deep learning has been applied to mathematical and physical sciences research [41], including solving partial differential equations [42,43] and performing Monte Carlo simulations [44,45]. In this knot type classification problem, we designed our neural networks under a *supervised learning* process: neural networks are trained on a dataset of samples with known classification, and the trained

*liangdai@cityu.edu.hk

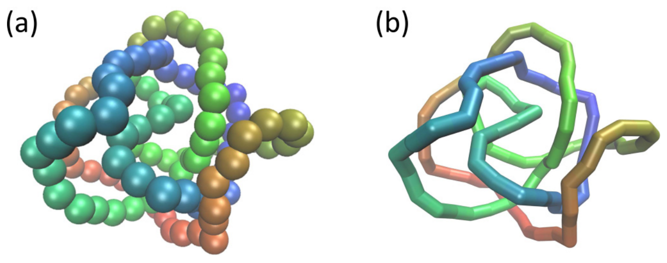

FIG. 1. Two visual representations of the same circular polymer conformation in simulation. The task of this work is to recognize knots in such a circular polymer conformation. (a) Touching-bead model of a circular polymer in simulation. (b) The representation of a polymer conformation by rods.

models are applied to recognize the class of unseen samples, thus achieving generalization to new data.

The motivation of this work is twofold. First, knot type is a *global* and *sequential* property that depends on the arrangement of monomer positions of a polymer. This property is different from other local properties of polymer conformations, e.g., polymer structures and monomer packing [46]. It is of great interest to examine whether neural networks are effective at addressing these global and sequential properties. There is active ongoing development of NN methods, and it is beneficial to find out which ones are most suitable for the knot classification problem. Second, knot classification is an important and unsolved problem in mathematical and physical sciences. If NNs can successfully classify knots, they may provide new insights into solving the knot classification problem.

II. METHODS AND MODEL ARCHITECTURE

A. Generation of polymer conformations with various knot types

We perform Monte Carlo simulations to sample equilibrium conformations of single circular polymers in confinement [47,48]. We model the polymer as a string of beads (Fig. 1). Bead-bead interactions are described by hard-core repulsion. The hardcore diameter D_{bead} equals the bond length, a, while a is used as the unit length. To generate sufficient numbers of knotted conformations for relatively short polymers, we sample polymer conformations in high-knotting-probability regimes. Accordingly, we impose a moderate bending potential and spherical confinement. A bending potential is applied to produce a persistence length of $4a$. For polymers with lengths of $L_{\text{polymer}} = 60$, 80, and 100, we use the surrounding spherical confinement of diameters of $D = 9a$, $10a$, and $11a$, respectively. The Monte Carlo simulation starts with a random circular conformation. In every Monte Carlo step, we perform a trial crankshaft move to update the conformation. Note that the crankshaft move is a global move, which can modify the knot type of the conformation. As a result, the knot type evolves during the simulation. We calculate the self-correlation time of the knot type, which is approximately 5.4×10^3 steps. Accordingly, we save one conformation every 10^4 steps so that the saved conformations are usually uncorrelated with each other. For every sampled polymer conformation, we analyze its knot type through the Alexander polynomials [49]. We sample 2×10^5 or 2×10^6 polymer conformations for each of five knot types: 0, 3_1, 4_1, 5_1, and 5_2. See examples of conformations in Fig. 2. More details about Monte Carlo simulations can be found in our previous studies [50–53] and in Appendix A.

B. Preprocessing of polymer conformations

The polymer conformation is represented by the trajectory of its monomer positions, i.e., XYZ coordinates of all monomers. The generated XYZ coordinates can have a wide spread of values. To improve NN performance, we convert the trajectory of positions to their relative increments from the previous coordinate (Fig. 3). For example, a trajectory of X-coordinates $[0, 0.8, 0.5, 1.4, 2.2]$ is converted to $[+0.8, -0.3, +0.9, +0.8]$. Since the bond length, i.e., the

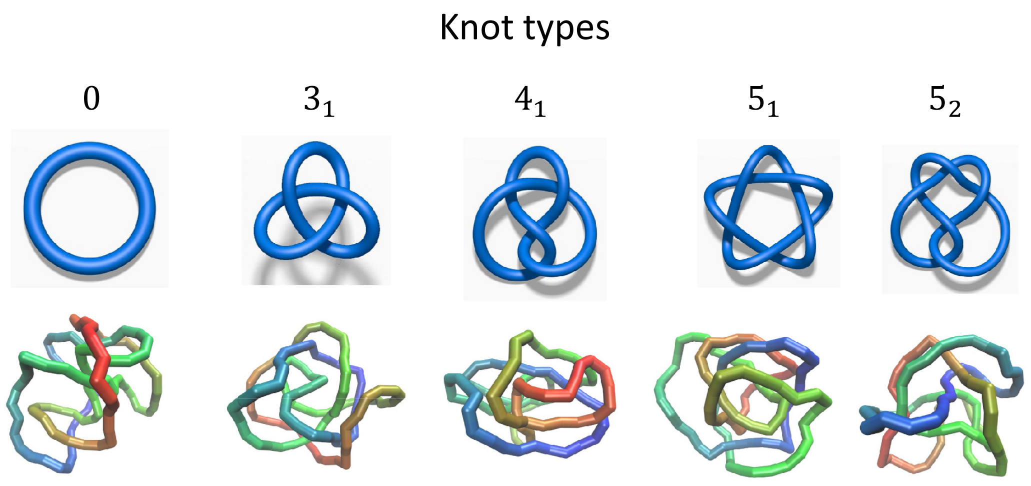

FIG. 2. Five knot types that are to be classified. The five diagrams at the top display the simple representations of these five knot types. The five diagrams at the bottom are examples of conformations generated by our Monte Carlo simulations. The knot types of these conformations are usually too complicated to be identified visually, and we train neural networks to recognize knots in these circular polymers.

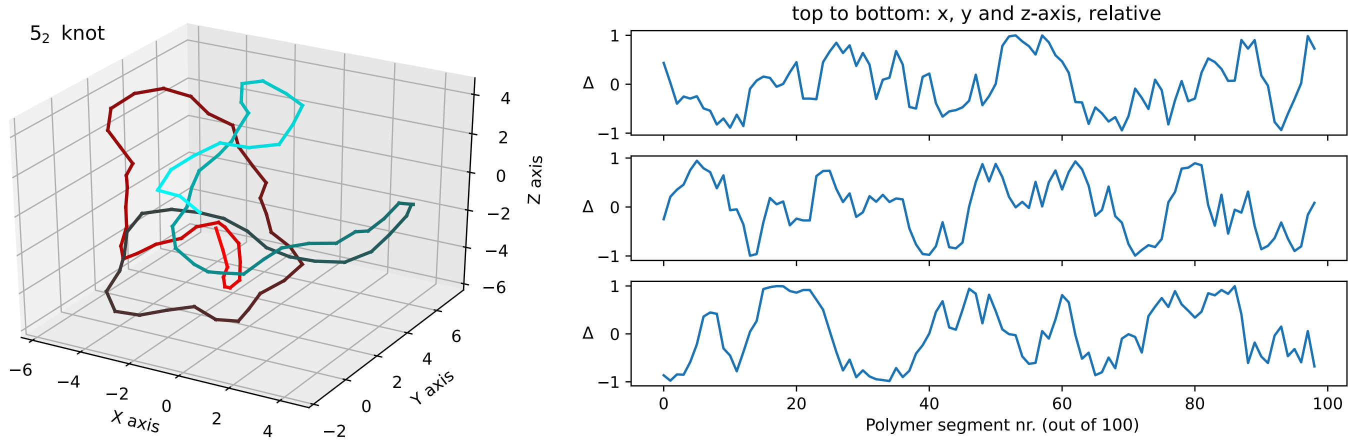

FIG. 3. Left: An example of polymer conformation with $L_{\text{polymer}} = 100$. Right: The inputs of this conformation for neural networks. Δx, Δy, Δz represent the changes in monomer positions in the x, y, and z directions at every step, e.g., from the fifth to the sixth monomer.

distance between two adjacent monomers, is a unit length, this conversion yields an additional benefit of scaling the data in the [0, 1] range, akin to normalization. Normalization is a common preprocessing technique that ensures that the input sequence has a stable mean and variance, which in turn prevents large error gradient values from accumulating, and prevents the training process from becoming numerically unstable [54].

C. Feed-forward neural network

Many NN architectures have been developed, with specific optimizations for the nature of the task at hand. To examine which NN architecture is more effective in knot classification, we test both a feed-forward neural network (FFNN) and a recurrent neural network (RNN). In a feed-forward neural network, input coordinates are flattened and routed to the first layer, from where the information flows unidirectionally until the output layer. In [46], the authors successfully use a nearly identical setup to recognize different properties of polymers. These models are conceptually simple and fast to train (Fig. 4).

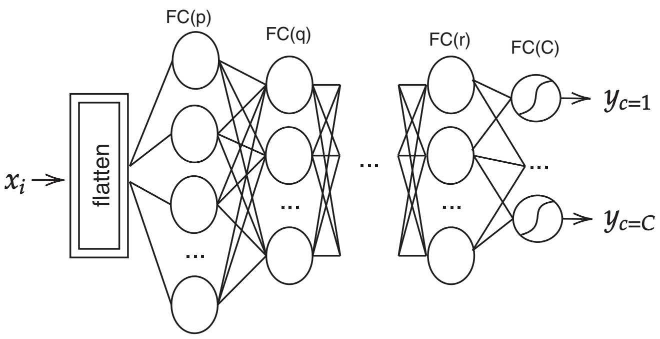

FIG. 4. The feed-forward architecture used in this work. 3D coordinates are flattened and sent through layers of $p, q, \ldots, r$ fully connected neurons, abbreviated FC(p), FC(q), FC(r), etc. The output layer contains sigmoid-activated neurons for C classes, performing the softmax operation to output class probabilities for y_c. Please see Appendix B for technical details on neural networks.

D. Recurrent neural network

Recurrent neural network (RNN) models are suited for data with an inherent order, especially time-series data, such as price movement in financial forecasting, sensor readings, but also natural language processing and machine translation, and sequence data in general [55]. In our case, knotting is a global and sequential property of the chain; the causality between each segment determines the knot type. This motivates us to treat it as an ordered sequence and experiment with recurrent neural networks.

We use a refined formulation of RNN called long short-term memory (LSTM) (Fig. 5) introduced by Hochreiter and Schmidhuber [56]. The number of cells in one LSTM layer of an NN equals the size of the representation, e.g., the polymer length in this work. LSTM cells carry an internal state (intuitively called *memory*), which captures the full history of preceding steps and tries to model their interdependencies. Information update in the state is controlled by gates that

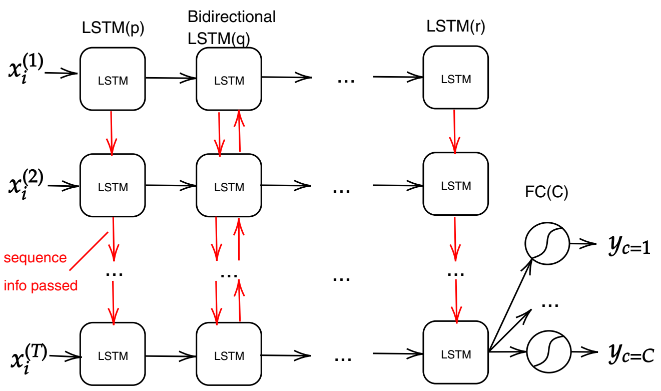

FIG. 5. LSTM-based recurrent neural network used in our work. Input $x_i^{(t)}$ represents the tth step of input sequence x_i. The first layer of LSTM(p) cells accepts three-dimensional inputs, and produces p-dimensional outputs, which are passed downstream. The LSTM(q) layer is bidirectional, which scans the sequence forward and backward. Data flow from layer to layer, as well as between cells within layers, capturing sequentiality. The last layer contains sigmoid neurons, outputting C class probabilities y_c, basing its predictions on the hidden state of the final recurrent layer's Tth cell.

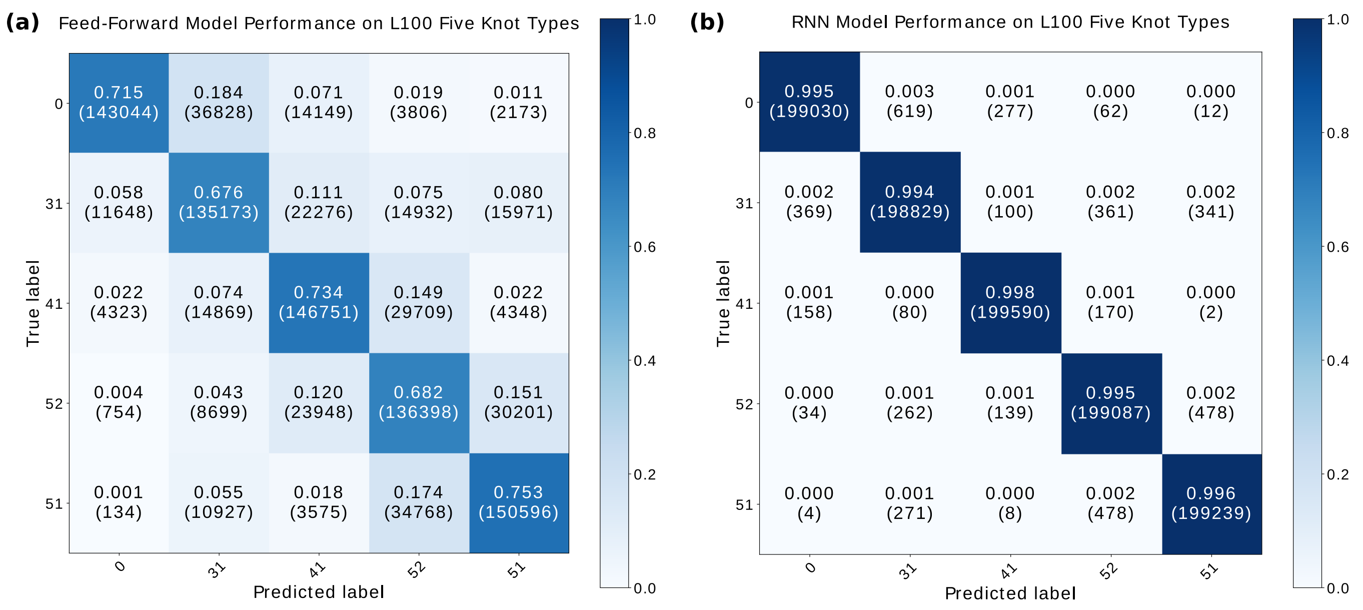

FIG. 6. Feed-forward and RNN models trained with polymers of length $L_{\mathrm{polymer}} = 100$ were used to predict five knot types on 1 million unseen test data. Predicted labels of the five knot types were plotted against true labels in a confusion matrix. The five knot type labels are knot 0, 3_1, 4_1, 5_1, and 5_2. The cells in the confusion matrix contain both the percentage of accurate predictions and the absolute number of data instances in parentheses. The colorbar represents percentage values from 0 to 1 with a blue colormap. (a) The feed-forward network is able to predict with accuracy between 67.6% and 75.3%; (b) the RNN model is able to classify the polymers to their correct knot types consistently at above 99%.

determine what to keep, what to forget, and what to pass on to the output. The term *gate* is used to show that information flowing in is regulated (via pointwise multiplication), i.e., input values are amplified or attenuated as needed. Gates are parametrized by trainable weights, which define the cell's behavior. Specifics are omitted in this explanation, which is aimed at giving a general intuition. Please see Appendix B for technical details on neural networks.

Bidirectional RNNs were introduced by Schuster and Paliwal [57] and appropriated for LSTM by Graves and Schmidhuber in 2005 [58]. The bidirectional layer is an important element as it sends information both forward and backward between sequence steps, effectively processing the sequence from both ends. We want to leverage this forward and backward context-awareness to improve the predictive power of the model when applied to knots in polymer chains, as the knot type is the same regardless of how the sequence is flipped.

It should be noted that the number of steps is not an intrinsic property of the model. This allows neural networks trained on one polymer length to be applied to other polymer lengths. Figure 5 shows the LSTM network in its unrolled form, where the vertical axis is expanded to match sequence steps, but the same trained cell is reused across the layer. For $L_{\mathrm{polymer}} = 100$, there would be 100 cells laid out vertically in each layer. Every next layer would introduce cells of differently sized hidden states, learning new representations and giving the network its depth. More on deep RNNs can be found in [59].

This dependence on adjacent steps of the sequence is lost in the FFNN model. Our approach contains stacked layers of LSTM cells, each taking a sequence in and outputting a transformed sequence for the next layer. The bidirectional layer sends information both forward and backward between the steps, effectively processing the polymer chain from both ends. LSTM cells support multivariate input at each sequence step, hence flattening is not needed. The same gradient descent process is utilized to find the parameters of LSTM cells.

E. Implementation of neural networks

We implement our networks using Keras [60] with a Tensorflow 2.0 backend [61,62] and GPU support. For model development, we randomly partition the corresponding dataset into 72%, 18%, and 10% portions as training data, holdout validation data, and test data, respectively. The model is trained using the training dataset, with the training progress monitored by the holdout validation set. Models that achieve the best performance on the validation set during training are saved later for evaluation on test data.

F. Evaluation of performance

We evaluate the performance of knot identification using the confusion matrix as shown in Figs. 6 and 7. In the confusion matrix, each column represents the instances of a predicted knot type while each row represents the ground truth knot type. We use accuracy as our main metric of success, defined as the sum of the diagonal elements divided by the total sum of the confusion matrix.

III. RESULTS AND DISCUSSION

A. Accuracy in knot identification

We train neural networks (NNs) to identify polymer knot conformations with the five knot types shown in Fig. 2.

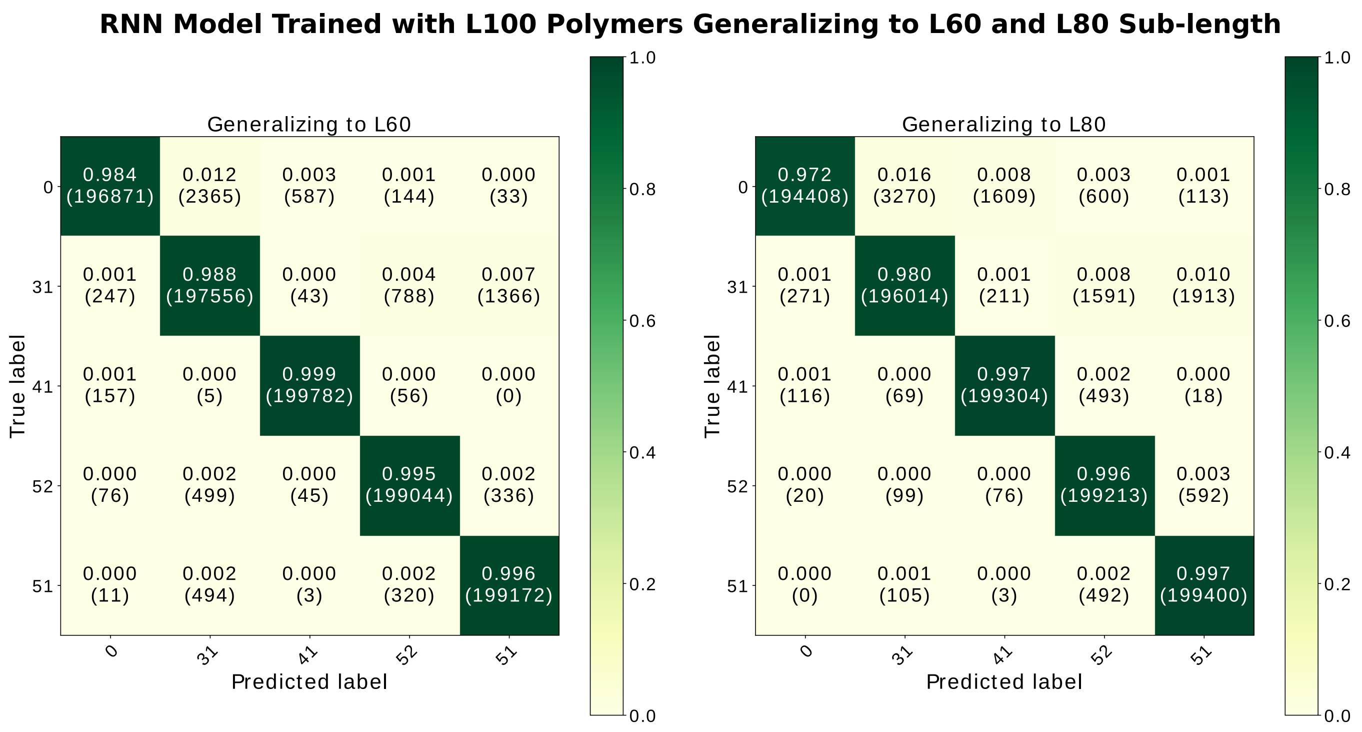

FIG. 7. The LSTM-based RNN model from Fig. 6(b) generalizing to predict knot types on polymers of variable length. The model was trained using polymer length $L_{\text{polymer}} = 100$ and was asked to predict knot type labels on 1 million $L_{\text{polymer}} = 60$ polymers and 1 million $L_{\text{polymer}} = 80$ polymers. Predicted labels of the five knot types were plotted against true labels in the confusion matrix. The five knot type labels are knot 0, 3_1, 4_1, 5_1, and 5_2. The cells in the confusion matrix contain both the percentage of accurate predictions and the absolute number of data instances in parentheses. The colorbar represents percentage values from 0 to 1 with a yellow-green colormap. Left is the confusion matrix for $L_{\text{polymer}} = 60$ and right is for $L_{\text{polymer}} = 80$, and both generalizations yield good accuracy.

The evaluation of trained NNs on unseen new test data is presented in Fig. 6, which contains two confusion matrices. The confusion matrices consist of both correct (diagonal) and wrong (off-diagonal) predictions of knot types. For the wrong predictions, we observe some bias, such as mislabeling between knots 5_1 and 5_2, and between unknots and knot 3_1. By tuning the NN architecture and increasing the dataset size, mislabeling becomes rare and we achieve an overall accuracy above 99%. For simplicity, we focus on the overall accuracy.

Table I displays the results obtained from various datasets and NN models. The accuracy strongly depends on the NN architecture, the training dataset size N_{data}, and the polymer length L_{polymer}. We discuss the dependence of the prediction accuracy on these factors in the following paragraphs.

First, accuracy depends on the NN architecture. We find that RNNs can achieve a substantially higher accuracy than FFNNs. For instance, in the case of $N_{\text{knot}} = 5$, $L_{\text{polymer}} = 100$, $N_{\text{data}} = 2 \times 10^6$, the RNN achieves an accuracy of 99.59%, compared to the feed-forward NN, which only reaches an accuracy of 71.25%. The better performance of the RNN in identifying the knot type is understandable for the following reasons. The knot type of a polymer depends on the overall

TABLE I. Polymer length, dataset size, and design of the neural network. Dataset size is the number of polymers per knot type. Models were trained using 72% of the dataset, with the training progress monitored on the 18% hold-out validation dataset. All models were trained to classify five knot types: knot 0, 3_1, 4_1, 5_1, and 5_2.

Polymer Length	Dataset size[a]	NN architecture	Number of parameters	Training accuracy	Validation accuracy
60	200 000	Feed-forward	428 685	81.62%	80.23%
60	200 000	Recurrent	862 625	99.17%	**98.51%**
80	200 000	Feed-forward	464 685	50.00%	43.38%
80	200 000	Recurrent	862 625	98.20%	**95.85%**
100	200 000	Feed-forward	2 073 185	48.73%	47.59%
100	2 000 000	Feed-forward	1 696 885	72.05%	71.25%[b]
100	200 000	Recurrent	729 415	98.15%	91.66%
100	2 000 000	Recurrent	889 385	99.90%	**99.59%**[c]

[a]Number of polymers per knot type, $N_{\text{knot}} = 5$.
[b]Test dataset evaluation results for this feed-forward model shown in Fig. 6(a).
[c]Test dataset evaluation results for this RNN model shown in Fig. 6(b).

shape, which manifests in the sequential order of monomer positions, not their individual values. As described in Sec. II, the LSTM cell is deliberately designed for learning sequential causation in ordered data.

Second, accuracy decreases as polymer length grows. For instance, in the case of the RNN, $N_{\rm data} = 2 \times 10^5$, and $N_{\rm knot} = 5$, accuracy decreases from 98.51% to 95.85% and 91.66% for polymer lengths of 60, 80, and 100, respectively. While the polymer length increases linearly, the conformation space grows rapidly, and accordingly the complexity of classification increases. As will be shown in the following section, the trained RNN for a longer polymer can be applied to a shorter polymer.

Lastly, accuracy improves with the training dataset size. As observed for the RNN, $N_{\rm knot} = 5$, $L_{\rm polymer} = 100$, accuracy goes from 91.66% to 99.59% as the dataset expands from 2×10^5 to 2×10^6 conformations. It is well known that the training set size limits the maximum performance and the generalization of a machine learning model. From the practical viewpoint of applying NNs in identifying knot types, it is useful to know how many conformations are needed in training to achieve a threshold accuracy for a given polymer length and the number of knot types.

B. Generalization of the RNN model to sublength polymers

A practical machine learning model should generalize well to new data outside of the training set. We have demonstrated the generalization power of the RNN model trained with $L_{\rm polymer} = 100$, as it successfully predicts knot types in unseen polymers of the same length. Here we present a broader generalization of the RNN model trained with $L_{\rm polymer} = 100$ to identify knots in sublength polymers, $L_{\rm polymer} = 60$ and 80, i.e., data of different length than the training data.

As mentioned in Sec. II D, RNN models accept sequence inputs of any length. Our LSTM-based RNN model from Fig. 6(b), trained on $L_{\rm polymer} = 100$ is, without any modification, capable of ingesting polymer conformations of $L_{\rm polymer} = 60$ and 80. We have tested it and compiled the results of our trials in confusion matrices in Fig. 7. The LSTM-based RNN model is able to predict with accuracy above 98% for length $L_{\rm polymer} = 60$, and accuracy above 97% for length $L_{\rm polymer} = 80$. The new inputs of sublength polymers differ considerably from what the network saw at training time, which further demonstrates the effectiveness of NNs in this problem.

C. Control experiment

One concern with a difficult-to-interpret system such as neural networks is that there is no direct indication of what high-level concepts are being learned. We were not certain whether our NN learns the real topological attributes of a knot, or if it is just memorizing the data or picking up features other than the knot type, e.g., conformational size. To clarify this issue to some extent, we design a control experiment. We randomly divide 2 million polymer conformations with the same knot type 3_1 into two equally sized groups, and we train a binary classifier using our RNN design. As shown in

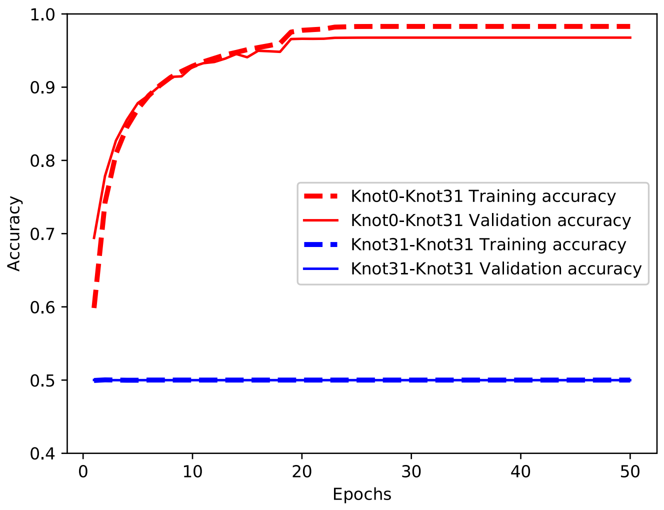

FIG. 8. Comparison of a real classification and a control experiment. Knot0-Knot31 refers to the data set of 1 million conformations of unknot and 1 million conformations of knot 3_1 (a real classification). Knot31-Knot31 refers to the data of 2 million conformations of knot 3_1 with two dummy labels (a control experiment). An epoch is an iteration over the entire training and validation data. Each epoch trains on 1 600 000 samples and validates on 400 000 samples.

Fig. 8, the NN is unable to start converging, at an accuracy of ∼50%, which is the baseline performance of a random guess. In contrast, the same NN achieves an above 95% accuracy for two groups of conformations with knot types of 0 and 3_1. Such a result suggests that no feature other than knot types of polymer conformations is learned by NN in the classification of conformations.

D. Interpretation of results

The feasibility of our RNN model to work not only on unseen, but also variable-length inputs, suggests that the learned abstractions are useful and sufficiently general. The knot's global and sequential properties are accounted for.

We find that two factors play important roles in achieving a high accuracy. First, LSTM-based RNN performs significantly better than FFNN. We conjecture that the multiple layers of our LSTM models learn a hierarchical representation of a polymer conformation as composed of smaller curves and shape artifacts, and they are sequenced on multiple levels. Second, feeding NN with the differences in positions between adjacent monomers performs better than feeding absolute positions of all monomers. It is likely that this transformation simplifies input space.

E. Computational efficiency

To evaluate the computational efficiency of the RNN, we compare the average computational time of determining knot types by the Alexander polynomials and the RNN. For polymer conformations with $L_{\rm polymer} = 100$, the average computational time of determining the knot type of one polymer by the Alexander polynomials is approximately 3×10^{-3} s, regardless of the knot type. For the same set of polymer conformations with $L_{\text{polymer}} = 100$, the average computational time of determining the knot type by NN is approximately 1.5×10^{-4} s, regardless of the knot type. So the RNN is about 20 times faster than the Alexander polynomials for these polymer conformations. Note that the RNN model is parallelized and runs on many GPU cores, while the computation of the Alexander polynomials runs on a single CPU core.

IV. CONCLUSION

In conclusion, this work demonstrates the effectiveness of NNs in identifying knot types of polymer conformations. Broadly speaking, knots represent a global and sequential property of polymer conformations, which are different from other local conformational properties [46]. This work provides insights on addressing global conformational properties using NNs, and it confirms that RNN is a more effective design for this problem.

Knot classification is still an unsolved problem, particularly for complex knots. In this work, the trained NN is only capable of identifying simple knots with limited polymer lengths. Hopefully, future studies can extend the capability, and eventually help solve the knot classification problem.

ACKNOWLEDGMENTS

This research is financially supported by City University of Hong Kong (Project No. 9610420), the National Natural Science Foundation of China (Project No. 21973080), Guangdong Basic and Applied Basic Research Fund (Project No. 2019A1515011736), and the Research Grants Council of Hong Kong Special Administrative Region, China (Projects No. 14309016 and No. 14301317).

O.V. and K.Y. contributed equally to this work.

APPENDIX A: MORE DETAILS ABOUT THE GENERATION OF POLYMER CONFORMATIONS

More details about the generation of polymer conformations can be found in Fig. 9.

APPENDIX B: NEURAL NETWORK DETAILS

Considering that it is an early stage for the applications of neural networks in physical science research, we present the basic knowledge of neural networks and some technical details in this Appendix, which should help others to reproduce our results and apply deep learning in their own research.

1. Building blocks of neural networks

On a high level, supervised learning for classification involves training a model f on data containing inputs $(x_1, x_2, \dots, x_N)$ and associated class labels $(y_1, y_2, \dots, y_N)$. It assumes that there is a continuity of operations defined by parameters θ that can be learned from data, and used to transform an input to the labeled output. It also defines a loss function $\mathcal{L}(y, \hat{y})$ that quantifies the error between the predicted label $\hat{y}$ and the true label y. Applying the model gives a prediction $\hat{y}_i = f(x_i|\theta)$. Training a model means finding parameters θ that minimize the loss $\mathcal{L}$ per data point.

The basic building block of a neural network is a neuron, depicted in Fig. 10. The parameters θ of the model thus become the weights $w = (w_1, w_2, \dots, w_n)$ and biases b of

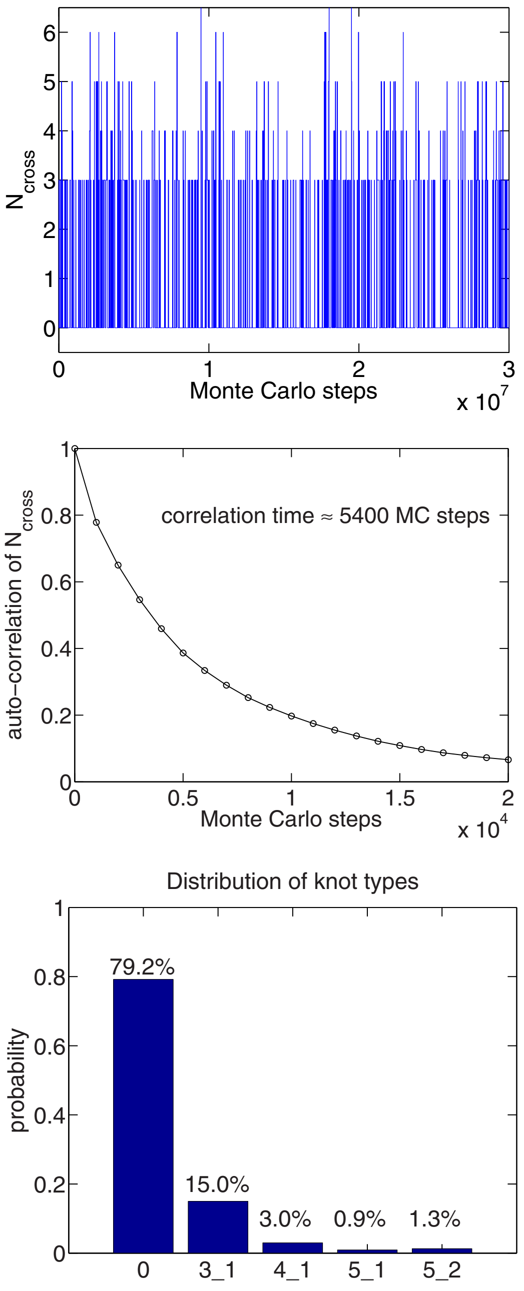

FIG. 9. Top: Evolution of the crossing number of the knot type, N_{cross}, during the Monte Carlo simulation. Here, N_{cross} represents the minimum crossing number for a knot type, e.g., 5 for the 5_2 knot. Middle: The autocorrelation of N_{cross} as a function of the simulation time. The correlation time is approximately 5.4×10^3 steps. Bottom: The distribution of knot types in our simulation of the polymer $L = 100$ and $L_p = 4a$ confined in a sphere with a diameter of $D = 11a$.

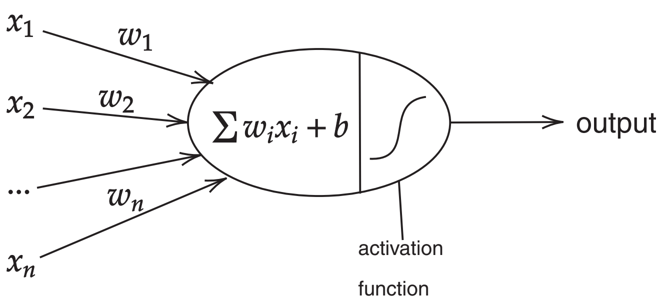

FIG. 10. The neuron calculates a weighted sum of the inputs, and passes the result through a nonlinear activation function $a(w^T x + b)$, to scale the output in the $[\mathbf{0}, \mathbf{1}]$ range. Many neurons are connected together in a neural network.

each neuron. The nonlinear activation function is an important part as it enables the network to approximate any function, as shown by Cybenko [63], who used the smooth sigmoid [Eq. (B1)] activation, while we use the computationally faster ReLU [Eq. (B2)]. The neuron's operation can thus be summarized as

$$\frac{1}{1+\exp(-\sum_i w_i x_i - b)} \quad \text{Sigmoid,} \tag{B1}$$

$$\max\left(0, \sum_i w_i x_i - b\right) \quad \text{ReLU.} \tag{B2}$$

Since each polymer 3D sequence can be categorized into one of the five knot type categories, this problem is treated as a multiclass classification problem [55], and *categorical cross-entropy or multiclass log loss* [64] [Eq. (B3)] is used as the loss function $\mathcal{L}$,

$$\mathcal{L}(y, \hat{y}) = -\sum_{c=1}^{M} \hat{y}_c \ln(y_c), \tag{B3}$$

where M is the number of class labels c, $\hat{y}_c$ is the predicted probability of the label belonging to class c, and y is true probability distribution over class labels. The cross-entropy-based loss function computes the distance between the predicted probability distribution of the labels and the ground-truth distribution.

2. Optimization of neural networks

Training is done by an optimizer, using an iterative process called gradient descent. The performance of the network is measured by the cost function C, which factors in the combined losses on all data points. An example cost function averages the losses, $C = 1/N \sum_{i=1}^{N} \mathcal{L}(y_i, \hat{y}_i)$. The optimizer makes small perturbations in θ across the network, in an optimization bid to minimize C. Since Eqs. (B1) and (B2) are a differentiable function, the whole network is end-to-end differentiable (for ReLU, the derivative at $x = 0$ is taken to be 0). This allows the optimizer to compute partial derivatives for each neuron, $\nabla C_w^n = \partial C/\partial w$ and $\nabla C_b^n = \partial C/\partial b$ for neuron n. The derivatives can be propagated backward from the output to the input via the backpropagation algorithm laid out in [65]. The modern computational implementation is highly optimized, but further details are outside the scope of this paper. The update of parameters in a training step can be expressed as

$$w \to w' = w - \eta \nabla C_w, \tag{B4}$$

$$b \to b' = b - \eta \nabla C_b, \tag{B5}$$

where η is the learning rate set by the optimizer. The training time and quality depend on the choice of the learning rate. We use optimization schemes that support adaptive learning rates such as *Adam* [66] and Hinton's *RMSprop* [67]. The model is considered trained when the weight updates become sufficiently small.

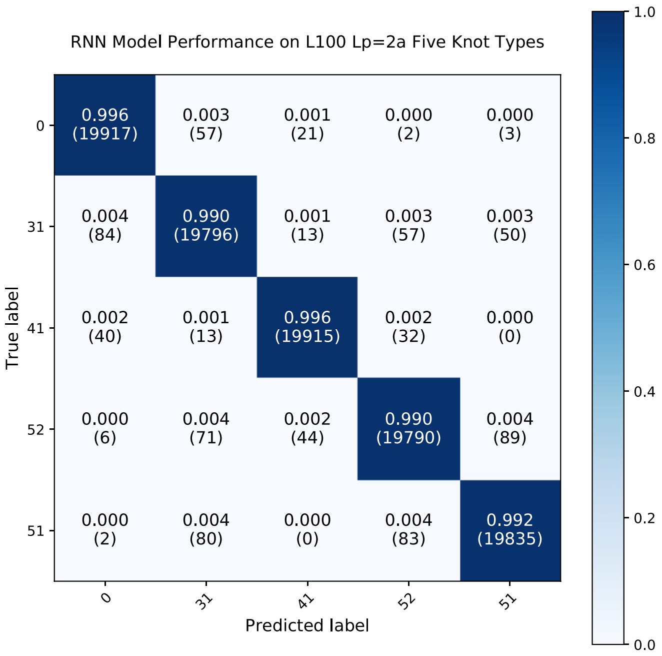

FIG. 11. The confusion matrix of applying the RNN model trained from the conformations of $L_p = 4a$ to the conformations with $L_p = 2a$.

APPENDIX C: NN ACCURACY FOR POLYMER CONFORMATIONS WITH A DIFFERENT BENDING STIFFNESS

In the body of the paper, the polymer conformations are generated with a persistence length $L_p = 4a$. To examine whether our NN also works for polymer conformations with a different bending stiffness, we generate 20 000 conformations of $L_{\text{polymer}} = 100$ and $L_p = 2a$ for each of the five knot types. Then, we apply the RNN model trained from conformations with $L_p = 4a$ to classify these new conformations with $L_p = 2a$. The prediction accuracy is above 99% for every knot type. The confusion matrix for predicting the $L_p = 2a$ polymers is shown in Fig. 11. These results suggest that the prediction accuracy of our NN is insensitive to the bending stiffness.

[1] E. Orlandini, Statics and dynamics of DNA knotting, J. Phys. A **51**, 053001 (2017).
[2] C. Micheletti, D. Marenduzzo, and E. Orlandini, Polymers with spatial or topological constraints: Theoretical and computational results, Phys. Rep. **504**, 1 (2011).
[3] V. V. Rybenkov, N. R. Cozzarelli, and A. V. Vologodskii, Probability of DNA knotting and the effective diameter of the DNA double helix, Proc. Natl. Acad. Sci. USA **90**, 5307 (1993).
[4] S. Y. Shaw and J. C. Wang, Knotting of a DNA chain during ring closure, Science **260**, 533 (1993).
[5] W. R. Taylor, A deeply knotted protein structure and how it might fold, Nature (London) **406**, 916 (2000).
[6] M. Jamroz, W. Niemyska, E. J. Rawdon, A. Stasiak, K. C. Millett, P. Sułkowski, and J. I. Sulkowska, Knotprot: A database of proteins with knots and slipknots, Nucl. Acids Res. **43**, D306 (2014).
[7] A. R. Klotz, B. W. Soh, and P. S. Doyle, Motion of Knots in DNA Stretched by Elongational Fields, Phys. Rev. Lett. **120**, 188003 (2018).
[8] J. Tang, N. Du, and P. S. Doyle, Compression and self-entanglement of single DNA molecules under uniform electric field, Proc. Natl. Acad. Sci. (USA) **108**, 16153 (2011).
[9] A. Narros, A. J. Moreno, and C. N. Likos, Effects of knots on ring polymers in solvents of varying quality, Macromolecules **46**, 3654 (2013).
[10] L. Dai, B. W. Soh, and P. S. Doyle, Effects of side chains on polymer knots, Macromolecules **52**, 6792 (2019).
[11] T. Christian, R. Sakaguchi, A. P. Perlinska, G. Lahoud, T. Ito, E. A. Taylor, S. Yokoyama, J. I. Sulkowska, and Y. Hou, Methyl transfer by substrate signaling from a knotted protein fold, Nat. Struct. Mol. Biol. **23**, 941 (2016).
[12] Á. San Martín, P. Rodriguez-Aliaga, J. A. Molina, A. Martin, C. Bustamante, and M. Baez, Knots can impair protein degradation by atp-dependent proteases, Proc. Natl. Acad. Sci. (USA) **114**, 9864 (2017).
[13] D. Marenduzzo, C. Micheletti, E. Orlandini *et al.*, Topological friction strongly affects viral DNA ejection, Proc. Natl. Acad. Sci. (USA) **110**, 20081 (2013).
[14] D. Marenduzzo, E. Orlandini, A. Stasiak, L. Tubiana, D. W. Sumners, L. Tubianae, and C. Micheletti, DNA–DNA interactions in bacteriophage capsids are responsible for the observed DNA knotting, Proc. Natl. Acad. Sci. (USA) **106**, 22269 (2009).
[15] L. Zhang, A. J. Stephens, A. L. Nussbaumer, J.-F. Lemonnier, P. Jurček, I. J. Vitorica-Yrezabal, and D. A. Leigh, Stereoselective synthesis of a composite knot with nine crossings, Nat. Chem. **10**, 1083 (2018).
[16] V. Marcos, A. J. Stephens, J. Jaramillo-Garcia, A. L. Nussbaumer, S. L. Woltering, A. Valero, J.-F. Lemonnier, I. J. Vitorica-Yrezabal, and D. A. Leigh, Allosteric initiation and regulation of catalysis with a molecular knot, Science **352**, 1555 (2016).
[17] I. Coluzza, P. D. J. van Oostrum, B. Capone, E. Reimhult, and C. Dellago, Sequence Controlled Self-Knotting Colloidal Patchy Polymers, Phys. Rev. Lett. **110**, 075501 (2013).
[18] G. Polles, D. Marenduzzo, E. Orlandini, and C. Micheletti, Self-assembling knots of controlled topology by designing the geometry of patchy templates, Nat. Commun. **6**, 6423 (2015).
[19] M. Marenda, E. Orlandini, and C. Micheletti, Discovering privileged topologies of molecular knots with self-assembling models, Nat. Commun. **9**, 3051 (2018).
[20] L. Zhang, J.-F. Lemonnier, A. Acocella, M. Calvaresi, F. Zerbetto, and D. A. Leigh, Effects of knot tightness at the molecular level, Proc. Natl. Acad. Sci. (USA) **116**, 2452 (2019).
[21] V. P. Patil, J. D. Sandt, M. Kolle, and J. Dunkel, Topological mechanics of knots and tangles, Science **367**, 71 (2020).
[22] C. Plesa, D. Verschueren, S. Pud, J. van der Torre, J. W. Ruitenberg, M. J. Witteveen, M. P. Jonsson, A. Y. Grosberg, Y. Rabin, and C. Dekker, Direct observation of DNA knots using a solid-state nanopore, Nat. Nanotech. **11**, 1093 (2016).
[23] A. Suma and C. Micheletti, Pore translocation of knotted DNA rings, Proc. Natl. Acad. Sci. (USA) **114**, E2991 (2017).
[24] R. K. Sharma, I. Agrawal, L. Dai, P. S. Doyle, and S. Garaj, Complex DNA knots detected with a nanopore sensor, Nat. Commun. **10**, 1 (2019).
[25] S. Amin, A. Khorshid, L. Zeng, P. Zimny, and W. Reisner, A nanofluidic knot factory based on compression of single DNA in nanochannels, Nat. Commun. **9**, 1506 (2018).
[26] A. Jain and K. D. Dorfman, Simulations of knotting of DNA during genome mapping, Biomicrofluidics **11**, 024117 (2017).
[27] J. G. Reifenberger, K. D. Dorfman, and H. Cao, Topological events in single molecules of *e. coli* DNA confined in nanochannels, Analyst **140**, 4887 (2015).
[28] This is the same Haken who solved the famous four-color map conjecture, jointly with Appel.
[29] W. Haken, Theorie der normalflächen, Acta Math. **105**, 245 (1961).
[30] J. W. Alexander, Topological invariants of knots and links, Trans. Am. Math. Soc. **30**, 275 (1928).
[31] V. F. R. Jones, Hecke algebra representations of braid groups and link polynomials, Ann. Math. **126**, 335 (1987).
[32] P. Freyd, D. Yetter, J. Hoste, W. R. Lickorish, K. Millett, and A. Ocneanu, A new polynomial invariant of knots and links, Bull. Am. Math. Soc. **12**, 239 (1985).
[33] P. Ozsváth and Z. Szabó, Holomorphic disks and knot invariants, Adv. Math. **186**, 58 (2004).
[34] M. Khovanov, A categorification of the jones polynomial, Duke Math. J. **101**, 359 (2000).
[35] These knot homology invariants can recognize some other knot types as well, e.g., the trefoil 3_1, figure-eight 4_1, etc.
[36] D. Michie, D. J. Spiegelhalter, C. Taylor *et al.*, Machine learning, Neural Stat. Class. **13**, 1 (1994).
[37] T. M. Mitchell, *Machine Learning*, 1st ed. (McGraw-Hill, New York, 1997).
[38] G. E. Hinton, S. Osindero, and Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural Comput. **18**, 1527 (2006).
[39] A. Krizhevsky, I. Sutskever, and G. E. Hinton, Imagenet classification with deep convolutional neural networks, Commun. ACM **60**, 84 (2017).
[40] Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, Nature (London) **521**, 436 (2015).
[41] J. Carrasquilla and R. G. Melko, Machine learning phases of matter, Nat. Phys. **13**, 431 (2017).
[42] E. Weinan, J. Han, and A. Jentzen, Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations, Commun. Math. Stat. **5**, 349 (2017).
[43] Q. Wei, Y. Jiang, and J. Z. Y. Chen, Machine-learning solver for modified diffusion equations, Phys. Rev. E **98**, 053304 (2018).
[44] L. Wang, Exploring cluster Monte Carlo updates with Boltzmann machines, Phys. Rev. E **96**, 051301(R) (2017).

[45] W. Yu, Y. Liu, Y. Chen, Y. Jiang, and J. Z. Chen, Generating the conformational properties of a polymer by the restricted Boltzmann machine, J. Chem. Phys. **151**, 031101 (2019).
[46] Q. Wei, R. G. Melko, and J. Z. Y. Chen, Identifying polymer states by machine learning, Phys. Rev. E **95**, 032504 (2017).
[47] L. Dai, J. J. Jones, J. R. van der Maarel, and P. S. Doyle, A systematic study of DNA conformation in slitlike confinement, Soft Matter **8**, 2972 (2012).
[48] L. Dai, D. R. Tree, J. R. C. van der Maarel, K. D. Dorfman, and P. S. Doyle, Revisiting Blob Theory for DNA Diffusivity in Slitlike Confinement, Phys. Rev. Lett. **110**, 168105 (2013).
[49] M. Frank-Kamenetskii and A. Vologodskii, Topological aspects of the physics of polymers: The theory and its biophysical applications, Sov. Phys. Usp. **24**, 679 (1981).
[50] L. Dai, C. B. Renner, and P. S. Doyle, Metastable tight knots in semiflexible chains, Macromolecules **47**, 6135 (2014).
[51] L. Dai, C. B. Renner, and P. S. Doyle, Origin of Metastable Knots in Single Flexible Chains, Phys. Rev. Lett. **114**, 037801 (2015).
[52] L. Dai, C. B. Renner, and P. S. Doyle, Metastable knots in confined semiflexible chains, Macromolecules **48**, 2812 (2015).
[53] L. Dai and P. S. Doyle, Universal knot spectra for confined polymers, Macromolecules **51**, 6327 (2018).
[54] C. M. Bishop, *Neural Networks for Pattern Recognition* (Oxford University Press, New York, 1995).
[55] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, Cambridge, MA, 2016).
[56] S. Hochreiter and J. Schmidhuber, Long short-term memory, Neural Comput. **9**, 1735 (1997).
[57] M. Schuster and K. K. Paliwal, Bidirectional recurrent neural networks, IEEE Trans. Signal Proc. **45**, 2673 (1997).
[58] A. Graves and J. Schmidhuber, Framewise phoneme classification with bidirectional lstm and other neural network architectures, Neural Netw. **18**, 602 (2005).
[59] R. Pascanu, Ç. Gülçehre, K. Cho, and Y. Bengio, How to construct deep recurrent neural networks, arXiv:1312.6026 (2013).
[60] F. Chollet *et al.*, Keras, https://keras.io (2015).
[61] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems (2015), software available from tensorflow.org.
[62] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, Tensorflow: A system for large-scale machine learning, in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)* (2016), pp. 265–283.
[63] G. Cybenko, Approximation by superpositions of a sigmoidal function, Math. Control Sign. Syst. **2**, 303 (1989).
[64] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, A tutorial on the cross-entropy method, Ann. Oper. Res. **134**, 19 (2005).
[65] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Neurocomputing: Foundations of Research* (MIT Press, Cambridge, MA, 1988), pp. 696–699.
[66] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980.
[67] M. C. Mukkamala and M. Hein, Variants of rmsprop and adagrad with logarithmic regret bounds, in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17 (JMLR.org, 2017), pp. 2545–2553.