Weighted-persistent-homology-based machine learning for RNA flexibility analysis

Chi Seng Pun,∗ Brandon Yung Sin Yong,† Kelin Xia‡

October 28, 2019

Abstract

With the great significance of biomolecular flexibility in biomolecular dynamics and function analysis, various experimental methods and theoretical models are developed. Experimentally, Debye-Waller factor, also known as B-factor, measures atomic mean-square displacement and is usually considered as an important measurement for flexibilities. Theoretically, elastic network models, Gaussian network model, flexibility-rigidity model, and other computational models, have been proposed for flexibility analysis by shedding light on the biomolecular inner topological structures. Recently, a topology-based machine learning model is proposed. By using the features from persistent homology, this model achieves remarkable high accuracy in protein B-factor prediction. Motivated by its success, we propose weighted-persistent-homology (WPH)-based machine learning (WPHML) models for RNA flexibility analysis. Our WPH is a newly-proposed model, which incorporate physical, chemical and biological information into topological measurements using a weight function. In particular, we use local persistent homology (LPH), which is not to consider the topology of a whole RNA structure, but to focus on the topological information of local regions. Our WPHML model is validated on a well-established RNA dataset, and numerical experiments show that our model can achieve a Pearson correlation coefficient up to 0.5822. The comparison with the previous sequence-information-based learning models shows that a consistent increase of accuracy by at least 10% is achieved in our current model.

Keywords: RNA chain, B-factor, Weighted persistent homology, Local persistent homology, Machine learning

∗School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore, cspun@ntu.edu.sg
†School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore
‡School of Physical and Mathematical Sciences and School of Biological Sciences, Nanyang Technological University, Singapore, xiakelin@ntu.edu.sg
INTRODUCTION

Biomolecular functions usually can be analyzed by their structure properties through quantitative structure-property relationship (QSPR) models (or quantitative structure-activity relationship (QSAR) models). Among all the structure properties, biomolecular flexibility is of unique importance, as it can be directly or indirectly measured by experimental tools. Debye-Waller factor or B-factor, which is the atomic mean-square displacement, provides a quantitative characterization of the flexibility and rigidity of biomolecular structures. With the strong relation between structure flexibility and functions, various theoretical and computational methods are proposed for the modeling of flexibility, including molecular dynamics (MD), normal mode analysis (NMA), graph theory, elastic network models (ENMs), Gaussian network model (GNM), anisotropic network model (ANM), local density model (LDM), local contact model (LCM), weighted contact number (WCN) model, molecular nonlinear dynamics, stochastic dynamics, flexibility-rigidity index (FRI), etc.

In these models, biomolecular structures are usually modeled as graphs or networks, and a deterministic relationship is established between experimental B-factors and certain network properties, such as node degree, centrality, pseudo-inverse Laplacian matrix, pseudo-inverse Hessian matrixes, etc.

Other than the above deterministic models, data-driven machine learning models are also considered in flexibility analysis, thanks to the accumulation of ever-increasing experimental data. In these learning models, biomolecular genetic, epigenetic, evolutional and structural information are extracted and used as features in machine learning models, such as support vector machine (SVM), random forest (RF), gradient boost tree (GBT), artificial neural network (ANN), etc. Among these models, an evolutional information based learning model has been used in RNA flexibility analysis. In this model, position-specific iterative basic local alignment search tool (PSI-BLAST) is considered for homologous sequence identification. For each sample, a position-specific scoring matrix (PSSM) profile is calculated. The properties of the matrix are used as feature vectors and fed into various machine learning models. A high Pearson correlation coefficient (PCC) value of 0.53 between the test and predicted B-factor values has been achieved. Further, a multiscale weighted
colored graphs (MWCGs) based learning model is proposed to blindly predict protein B-factors. These MWCGs provide a series of graph features, that characterize the intrinsic flexibility of protein structure very well. The model can be used in the blind prediction of protein B-factor with a high accuracy, i.e., PCC value 0.66.

More recently, a persistent-homology (PH)-based machine learning model is proposed. In this model, persistent homology, which is a tool for data simplification and dimension reduction, is used for protein structure featurization. Different from conventional topology tools, which tend to oversimplify the structural information thus can only be used in qualitative modeling, persistent homology manages to retain the important geometric properties through a filtration process. Essentially, a series of simplicial complexes are generated and their topological information is characterized by homology groups. The “birth” and “death” of these homology generators, which preserve the geometric information, are recorded in persistent diagrams or barcode representation. Further, atom-specific PH and element-specific PH are considered in the model to classify the structures into different point sets with more detailed structure information. Moreover, two types of matrixes, based on Euclidean distance and multiscale interaction, are considered. Topological features are extracted from their corresponding barcodes and then combined with machine learning models. A PCC value up to 0.73 is achieved for a data-set with 364 proteins.

Motivated by the great success of the PH-based machine learning models in protein B-factor prediction. We propose weighted-persistent-homology (WPH)-based machine learning (WPHML) models for RNA B-factor prediction. Weighted persistent homology incorporates physical, chemical and biological information into the topological measurements with a weight function. In general, different weights can be assigned to points (0-simplices), edges (1-simplexes), triangles (2-simplexes), tetrahedrons (3-simplexes), etc. By assigning different weight values of 1 and 0 to points, we can naturally arrived at local persistent homology model and element-specific persistent homology model. Similarly, by using element interactions as weighted on edges, an interactive persistent homology is delivered. More importantly, a weighted boundary operator can be designed to embed further high-level relations into topological invariants. In the this paper, we only consider weight values on nodes, i.e., atoms, to select a local region around certain interested atoms, whose flexibility is to be
evaluated. PH analysis is then applied to the selected atoms. Features will be generated from the corresponding barcodes and then input into learning models. To test our models, we use the same dataset and same data preprocessing as in paper\(^{27}\). Our results show that WPH-based learning models can consistently outperform this sequence-based model in RNA B-factor prediction\(^{27}\), which again demonstrates the great importance of featurization and feature engineering in material, chemical and biological learning models.

The paper is organized as follows. Weighted persistent homology based featurization and the combination with different types of machine learning approaches are introduced in Section “Methodology”. In Section “Results”, we present the findings of our numerical results, including the comparison between the benchmark and our WPHML approaches and the sensitivity analysis of the model settings. The paper ends with a conclusion.

**METHODOLOGIES**

In this section, we give a brief introduction of persistent homology and weighted persistent homology firstly. Then, topology-based featurization is discussed in great details. After that, we briefly discuss three learning models that we consider.

**Topology-based feature engineering**

Data-driven sciences are widely regarded as the fourth paradigm that can fundamentally change sciences and pave the way for a new industrial revolution\(^{33}\). The past decade has witnessed the booming of various learning models in areas, such as data mining, natural language processing, image analysis, animation, visualization, etc. Gigantic progresses in these areas have been made. In contrast, the application of learning models in materials, chemistry and biology is far behind this trend. One of the most important reasons is featurization or feature engineering\(^{34–36}\). Compared with text, image or audio data, molecular structural data from material, chemistry and biology are highly irregular and differ greatly from each other. Essentially, different molecules can have not only different atom numbers or atom types, but also very different and complicated spatial connectivity. The structure complexity and high data dimensionality has significantly hampered the direct application of
learning models in these systems. To solve the problems, various ways of featurization have been proposed and a series of molecular descriptors (features) are generated. In general, molecular descriptors can be divided into three groups, structural measurements, physical measurements and genetic features. Structural measurements come from structural geometry, chemical conformation, chemical graph, structure topology, etc. Physical descriptors come from molecular formula, hydrophobicity, steric properties, and electronic properties, etc. Genetic features can be derived from gene sequences, gene expression, genetic interaction, evolution information, epigenetic information, etc.

Recently, persistent homology has been used in molecular characterization. With the unique attribute that balances geometric complexity and topological simplification, PH provides a unique structure featurization, that can be naturally combined with machine learning models. PH-based learning models have been successfully used in various aspect of drug design, including protein-ligand binding affinity prediction, solubility, toxicity, and partition coefficient. More recently, PH-based learning models are used in protein B-factor blind prediction and a remarkable high accuracy is obtained. These great success have inspired us to propose the WPHML for RNA B-factor prediction. To have a better understanding of our WPHML, a brief introduction of PH and WPH is given below.

**Persistent homology**

General speaking, persistent homology can be analyzed from three aspects, i.e., graph and simplicial complex, geometric measurements and topological invariants, and bridge between geometry and topology.

**Graph and simplicial complex** Graphs and networks, composed of only vertices and edges, are special cases of simplicial complexes. Geometrically, a 0-simplex is a vertex, a 1-simplex is an edge, a 2-simplex is a triangle, and a 3-simplex represents a tetrahedron. Simplices are the building block for the simplicial complex. In general, a simplicial complex $K$ is a finite set of simplices that satisfy two essential conditions. Firstly, any face of a simplex in $K$ is also in $K$. Secondly, the intersection of any two simplices in $K$ is either empty or shares faces.
Geometric measurements and topological invariants  Geometry models consider geometric information, such as coordinates, distances, angles, areas, various curvatures, vector bundles, etc. Graph models, study measurements such as degree, shortest path, clique, cluster coefficient, closeness, centrality, betweenness, Cheeger constant, modularity, graph Laplacian, graph spectral, Erdős number, percolation, etc. These geometric and graph descriptors characterize local and non-intrinsic information very well. In contrast, PH explores the intrinsic connectivity information measured by Betti number, which is a type of topological invariants that is unchanged under deformation. Geometrically, we can regard $\beta_0$ as the number of isolated components, $\beta_1$ the number of one-dimensional loops, circles, or tunnels and $\beta_2$ the number of two-dimensional voids or holes.

Bridge between geometry and topology  Different from either geometry or topology models, PH manages to incorporate geometric measurements into topological invariants, thus provides a balance between geometric complexity and topological simplification. The key idea of PH is a process called filtration. By varying the value of a filtration parameter, a series of simplicial complexes are generated. These nested simplicial complexes encode topological information of a structure from different scales. Some topological invariants “live longer” in these simplicial complexes, whereas others disappear very quickly when the filtration value changes. In this way, topological invariants can be quantified by their “lifespans” or “persisting times”, which are directly related to geometric properties. A persistent barcode can be generated from the birth, death and persistence of the topological invariants of the given dataset. An example of persistent barcode can be found in Figure 1.

Weighted persistent homology

Recently, we have systematically studied weighted PH models and their applications in biomolecular data analysis. General speaking, we can define weight values, which represents physical, chemical and biological properties, on vertices (atom centers), edges (bonds), or higher order simplexes (motif or domains). That is to say weighted PH can be characterized into three major categories, vertex-weighted, edge-weighted, and general-simplex-weighted models. These weighted values can be viewed as certain distance
measurements, and PH analysis can be applied. In this way, these properties are naturally incorporated into topological measurements. On the other hand, we can define a weighted boundary map, which can embed deeper interaction relationships into topology. Note that to ensure the consistence of the homology definition, weight values on different simplexes need to satisfy certain constraints.\textsuperscript{38,47,48} Previous PH models, including element specific PH\textsuperscript{45,46,49} and local persistent homology\textsuperscript{37} can be regarded as special cases of vertex-weighted PH. The multi-level PH, interactive PH, and electrostatic persistence\textsuperscript{46} are essentially edge-weighted PH.

In this paper, LPH is used for RNA local structure characterization. Biologically, an RNA chain is made up of a set of nucleotides, in which the size of the set of nucleotides can range from a low tens to a few thousands and above. In our LPH model, only atoms which are located within a specific Euclidean cut-off distance $E$ from each C1 atom in each chain are considered. Note that only the B-factor for C1 atoms are evaluated and compared with experimental data, the same way as in the paper\textsuperscript{27}. On the other hand, a nucleotide constitutes of heavy atoms C, N, O, and P. In our LPH, we consider the localized element-specific PH by using each type of the elements individually. Note that for each type of elements we selected, the central C1 atom is always included. Their topological features are drastically different from one another as shown in Figure 1. In deed, ESPH is capable of retaining crucial biological information during topological simplification\textsuperscript{50}.

**Topological features representation**

Results from PH analysis are pairs of “birth” and “death” values for different dimensions of Betti numbers. They can be represented as persistent barcodes or persistent diagrams. However, PH results are notorious for meaningful metric definition and statistic interface. Various methods are proposed\textsuperscript{51}, including barcode statistics, Tropical coordinates, binning approach, persistent image, persistent landscapes, image representations, etc.

In this paper, we only consider topological features constructed using a binning approach\textsuperscript{51}. More specifically, the filtration interval $[0, F]$ is divided into $N$ bins with an equal size, denoted as $f$. The number of barcodes which are located on each bin are then counted.
and used as feature vector. More specifically, the feature vector is defined as

\[ V_i = ||\{(b_j, d_j) \in B(\alpha, D) | b_j \leq iF/n \leq d_j\}||, 1 \leq i \leq N \]

where \( ||\cdot|| \) is cardinality i.e., the number of elements, of sets. Here \( b_j, d_j \) are referring to birth and death of bar \( j \). \( B(\alpha, D) \) is referring to the collections of barcodes with \( \alpha \) referring to the selection of atoms and \( D \) referring to the dimension of the Betti numbers. Essentially, for each C1 atom, we have an \( N \times 1 \) topological vector. Both one-element-type situation and four-element-type situation, in which all four topological vectors are combined together, are considered.

**Machine learning models**

After the topological features are represented in a structured feature vector, it can serve as input to predict the output of B-factor values with ML algorithms. We consider four main ML models, namely regularized linear regression, tree-based methods (including random forest and extreme gradient boosting), support vector regression, and artificial neural networks. All our ML algorithms are implemented in Python and thus the packages mentioned below refer to the packages in Python.

In the following descriptions of the ML models, we assume that we train our models with \( n \) data \( \{(x_i, y_i)\}_{i=1}^{n} \), where \( y_i \in \mathbb{R} \) is the normalized B-factor value of the \( i \)th sample (details of B-factor normalization will be discussed in Section Results), \( x_i \in \mathbb{R}^p \) is the structured topological feature vector of the \( i \)th sample, and \( p \) is the number of structured features. Conventionally, we denote by \( \hat{y} \) the predicted normalized B-factor value of a sample.

**Regularized linear regression**

Linear regression is a straightforward yet efficient approach to model the relationship between a quantitative response and features. Its incorporation with regularization can effectively address the high-dimensional setting where the number of features is larger than the sample size. The variable selection feature of the regularized linear regression makes it particularly suitable for our task as our feature vector is usually lengthy. The general formulation of
regularized linear regression can be read as the following regularized minimization problem:

$$\min_{\beta_0 \in \mathbb{R}, \beta \in \mathbb{R}^p} \sum_{i=1}^{n} (y_i - \beta_0 - x_i^\top \beta)^2 + R_\alpha(\beta), \tag{1}$$

where $R_\alpha(\cdot)$ is a regularization term. Once we obtain the minimizer of (1), denoted by $(\hat{\beta}_0, \hat{\beta})$, we predict the B-factor value of the test data with structured feature vector $x$ by $\hat{y} = \hat{\beta}_0 + x^\top \hat{\beta}$. The specification of $R_\alpha$ determines the shrinkage of $\hat{\beta}$ and statistical accuracy of $\hat{y}$.54–58

In our study, we consider the two typical choices of $R_\alpha$, namely L2-norm $(\alpha \|\beta\|_2^2)$ and L1-norm $(\alpha \|\beta\|_1)$, where $\alpha$ is the tuning parameter that strikes the balance between efficiency and regularization. The regression problem with these two types of regularization are also known as Ridge regression54 and least absolute shrinkage and selection operator (LASSO)55, respectively. The advantage of LASSO over Ridge regression is its variable selection feature, which has strong interpretable power. From the LASSO results, one can tell which part of structural information of the element is important. Both Ridge regression and LASSO are implemented with the package “scikit-learn”59.

**Tree-based methods**

Classification And Regression Tree (CART)60 or decision tree learning is a common method used in ML. Many variations of trees have been proposed with the pruning and ensemble methods. The simple and interpretable tree-based methods have the advantage of handling high-dimensional data without further adjustments. It addresses our concern with the lengthy feature vector deduced from topological feature representation. Among many candidates of tree-based methods, we consider Random Forest (RF)61,62 and Extreme Gradient Boosting (XGBoost).63

**RF** is an ensemble learning method that creates a variety of decision (regression) trees independently during training, where each decision tree is constructed using a random subset of the features as split candidates. During training of each tree, the split at each node is determined by the least-square method. In other words, for each region of each tree, we predict the B-factor value with the average of the B-factor values of the samples fallen in
the region. In a regression RF, the final prediction is the average of the predicted values of all individual samples. In the implementation of ensemble trees, the number of trees, minimum number of samples at each leaf node, and the number of split candidates in each splitting, i.e., parameter $mtry$, are all tuning parameters. In our application of RF, we choose $mtry = \lceil \sqrt{p} \rceil$ following Breiman and tune the other two hyperparameters. The RF is also implemented with the package "scikit-learn".

**XGBoost** has been one of the popular ML tools used by the winning teams of many ML challenges and competitions, such as the Netflix prize and various Kaggle challenges. Instead of computing the average output of all the individual trees as in a regression RF, each tree in XGBoost contributes a certain value which are added up iteratively. Such an additive training or gradient boosting allows the predicted values to approach the actual values as closely as possible. In our study, we tune the number of trees and the maximum tree depth, which affects the number of leaves in the trees, while the remaining parameters set default as defined by XGBoost. XGBoost is implemented with the package “xgboost”.

**Support vector regression**

Support vector regression (SVR), as a version of the well-known support vector machine (SVM) for regression, is another popular ML algorithm. The goal of an SVM model is to find a function $\beta_0 + x^T \beta$ that has at most $\epsilon$ deviation from the actual target values $y_i$ for all the training data while trying to be as flat as possible. Sometimes, the convex optimization problem is not feasible and a “soft margin” loss function is introduced. The SVR model $((\beta_0, \beta))$ is determined by the following minimization problem:

$$\min_{\beta_0, \beta, \xi, \xi^*} \frac{1}{2} \| \beta \|^2 + C \sum_{i=1}^n (\xi_i + \xi_i^*) \quad s.t. \quad \begin{cases} y_i - \beta_0 - x_i^T \beta_i & \leq \epsilon + \xi_i \\ \beta_0 + x_i^T \beta_i - y_i & \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* & \geq 0 \end{cases}$$

where $\xi$ and $\xi^*$ are slack variables to cope with the otherwise infeasible constraints of the optimization problem and the hyperparameter $C$ determines the trade-off between the efficiency and the amount up to which deviation larger than $\epsilon$ is tolerable. Typically, we adopt kernel methods to transform the input features from a lower to a higher dimensional space.
where the linear fit is suitable. Common choices of kernels include polynomial kernel, Gaussian kernel, and radial basis function (RBF) kernel. In our study, we have opted to use RBF kernel, i.e., \( K(x, x') = \exp(-\gamma \|x - x'\|_2^2) \), in our SVR model. The SVR is implemented with the package “scikit-learn”.

**Artificial neural network**

Artificial Neural Network (ANN) has been proved to be capable of learning to recognize patterns or categorize input data after training on a set of sample data from the domain. The ability to learn through training and to generalize broad categories from specific examples is the unique intelligence for ANN. Different from other ML algorithms, ANN requires the user to determine the architecture of the network, such as, the number of hidden layers, the number of nodes, and the specification of activation function in each layer. The hidden layers in ANN architecture allows the ANN to deal with nonlinear and complex problems more robustly and therefore can operate on more interesting problems. The number of hidden layers enables a trade-off between smoothness and closeness of fit. The number of nodes within a hidden layer determines the trade off between training time and training accuracy. The weights of each layer are optimized via the use of a learning algorithm called “backpropagation”. Since the ANN will involve the learning of a vast amount of weights, from the statistical perspective, overfitting problem arises. We adopt a recently proposed regularization technique called “dropout”, which is empirically proven magical. This approach also addresses the curse of dimensionality due to lengthy topological feature vector in our study.

In our study, the number of hidden layers, number of nodes in each hidden layer, and number of epochs are treated as hyperparameters. The hidden and output activation functions are set as sigmoid and leaky ReLU functions, respectively. Dropout rate is set to 20% and the remaining hyperparameters are set to default values. ANN is implemented with the package “keras.”
RESULTS

RNA dataset and data preprocessing

RNA dataset  We consider the same RNA data set and data preprocessing by Guruge et al.27. The chains are randomly split in the same manner with 75% of the chains go into training set and 25% go into test set27. The B-factor of each nucleotide is represented by its C1 atom. Stated differently, only B-factors for C1 atoms are considered.

B-factor normalization and outlier detection  The values of B-factors may differ significantly from chain to chain due to reasons such as relatively small number of residues in a protein chain or differences in refinement methods used75. Thus, the B-factors of each chain are normalized to have zero mean and unit variance27. The range of normalized B-factor falls approximately between -3.00 and 4.00. Further, before the raw B-factors are normalized, outliers values are first detected and removed using a median-based approach76. This is to eliminate raw B-factor values that are located on the extreme ends of the distribution.

Hyperparameter tuning and model setting

In our model, cut-off distance $E$, $F/E$ ratio, and bin size $f$ are parameters to be optimized. Normally, we choose the value of $E$ to range from 10 Å to 45 Å with a stepsize of 5 Å, i.e., $E = \{10 \text{ Å}, 15 \text{ Å}, 20 \text{ Å}, 25 \text{ Å}, 30 \text{ Å}, 35 \text{ Å}, 40 \text{ Å}, 45 \text{ Å}\}$. The filtration interval $F$ is defined such that the ratio of $F/E$ is between 0.5 to 1.0 with a stepsize of 0.1, $F/E = \{0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$. Both GUDHI77 and Dionysus78 packages are used. The persistent barcodes are generated based on the Vietoris-Rips complex.

To determine the optimal hyperparameter values for ML models, we conduct a five-fold cross validation (CV) using the training data. Specifically, the training data is randomly divided into five folds with similar number of chains. In each fold, for each combination of the hyperparameters, we find the predicted B-factor values for the left-out training data with the ML model trained by the remaining training data. The optimal hyperparameter set maximizes the out-of-sample PCC between the predicted and actual values across all folds. The optimal hyperparameters for each ML model can be found in Table 3.
We test the optimized trained model on the test set. Once the hyperparameter values of the dataset and models have been optimized, the trained models are evaluated using a test set that was non-overlapping with the training set. The PCC between the predicted and actual normalized B-factor values in the test set is calculated for each model

\[
PCC(y_i, \hat{y}_i) = \frac{\sum_{i=1}^{n}(y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})}{\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2 \sum_{i=1}^{n}(\hat{y}_i - \bar{\hat{y}})^2}},
\]

where \(\hat{y}_i\) is the predicted \(i\)-th B-factor value, \(\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i\) and \(\bar{\hat{y}} = \frac{1}{n} \sum_{i=1}^{n} \hat{y}_i\).

**Performance of WPHML**

Table 1 shows the best performance achieved by each ML model on test set. The best performance reported by Guruge et al.\(^27\) is used as a benchmark performance. The conditions in which the best test performances are obtained from can be found in Table 2 in Appendix.

For both single-element models and four-element-combined models, it can be seen that WPHML models are able to consistently outperform the evolution-based method (PSSM) by approximately at least 10% with only the exception of linear regression models (Ridge and LASSO). Among all the models, RF achieves the best result, i.e., PCC=0.5788. Moreover, the performance of RF model further improves to 0.5822 when the topological features for all four elements were used, which is about 15.8% increase.

The comparison between the results from single-element models and four-element-combined models shows that generally there is no significant improvement. In fact, SVM improves only slightly (approximately 0.8%), while XGBoost and ANN model even show some small reduction of accuracy. The results seem to be different from previous findings, that element-specific models always deliver better results.\(^{28,44,45,49}\) Note that previous models are based on protein structures. Comparably speaking, RNA structures are more regular and relatively simple. Similar topological features may be embedded in different types of element models. In this way, the additional features do not incorporate new information, instead they will contribute more noises, which causes the drop in performances. Noted that the best test performance of all the models except linear regression using a single element are all based on the element P.
Effect of Euclidean cut-off distance  Figure 2 shows the effect of cut-off distance. It can be seen that the PCCs of the fivefold cross validation using the topological features from both element P and all four elements, gradually improve and eventually plateaus off at approximately 35 Å. Note that 35 Å is larger than the generally-used cut-off distance in Gaussian network model, anisotropic network model, and other graph based models, which are usually around 8Å to 20 Å. One of the reason that larger cut-off distance delivers good results is that our predicted PCC values are predominantly determined by the several larger-sized RNAs. From Table 5, it can be seen that, even though our RF model has a fairly good correlation (0.5822), when the PCC for each individual chain in the test set is calculated, there is a wide range of distribution which ranges from -0.50 to 0.80. Moreover, although only 4 out of the 34 chains in the test set have a chain PCC higher than the overall PCC by RF, approximately 70% of the test data points come from these 4 chains. With that said, the performance of the test set is heavily based on these 4 chains. As long as the predictions on these 70% data points continue to improve, the overall performance of the model would continue to improve although there may be a reduction in performance on the remaining 30% of data points. This indicates that the evaluation method may have certain limitations. However, for a fair comparison, we still use it in the current paper.

Effect of F/E ratio  Figure 3 shows the effect of F/E ratio on the fivefold cross validation performance. At a low cut-off distance, the improvement in the fivefold cross validation performance improves more significantly when F/E ratio increases from 0.5 to 0.7. Beyond 0.7, the improvement in performance is very minimal. However, at a large cut-off distance, the performance is rather consistent from 0.5 to 1.0. This shows that the F/E ratio is not a significant hyperparameter to generate the dataset and it is more than sufficient to use an F/E ratio of 0.5 so as to minimize the number of unnecessary features generated especially as a large cut-off distance is required as discussed previously.

Effect of bin size  Figure 4 shows the changes in five-fold CV performance with respect to bin size. As the bin size decreases from 1.5 Å to 0.15 Å, the performance improves for all Euclidean cut-off distance. This indicates that with a smaller bin size, the finer details of
topological features are detected especially topological invariants that exist for a very short moment. The geometric information, embedded in the topological invariants, are key to the success of WPHML models.

**CONCLUSION**

In this paper, we propose the weighted-persistent-homology-based machine learning (W-PHML) models and use them in the RNA B-factor prediction. We found that our WPHML models can consistently deliver a better accuracy than the evolution-based learning models. In particular, local persistent homology and element-specific persistent homology are considered for topological feature generation. These topological feature based random forest model can deliver a PCC up to 0.5822, which is 15% increase of the accuracy than the previous model. Our WPHML models are suitable for any biomolecular structure based data analysis.

**ACKNOWLEDGMENTS**

This research is partially supported by Nanyang Technological University Startup Grants M4081840 and M4081842, Data Science and Artificial Intelligence Research Centre@NTU M4082115, and Singapore Ministry of Education Academic Research Fund Tier 1 RG31/18, Tier 2 MOE2018-T2-1-033.
References

1. J. A. McCammon, B. R. Gelin, and M. Karplus, Nature 267, 585 (1977).

2. N. Go, T. Noguti, and T. Nishikawa, Proc. Natl. Acad. Sci. 80, 3696 (1983).

3. M. Tasumi, H. Takenchi, S. Ataka, A. M. Dwidedi, and S. Krimm, Biopolymers 21, 711 (1982).

4. B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. States, S. Swaminathan, and M. Karplus, J. Comput. Chem. 4, 187 (1983).

5. M. Levitt, C. Sander, and P. S. Stern, J. Mol. Biol. 181, 423 (1985).

6. D. J. Jacobs, A. J. Rader, L. A. Kuhn, and M. F. Thorpe, Proteins-Structure, Function, and Genetics 44, 150 (2001).

7. I. Bahar, A. R. Atilgan, and B. Erman, Folding and Design 2, 173 (1997).

8. I. Bahar, A. R. Atilgan, M. C. Demirel, and B. Erman, Phys. Rev. Lett 80, 2733 (1998).

9. A. R. Atilgan, S. R. Durrell, R. L. Jernigan, M. C. Demirel, O. Keskin, and I. Bahar, Biophys. J. 80, 505 (2001).

10. K. Hinsen, Proteins 33, 417 (1998).

11. F. Tama and Y. H. Sanejouand, Protein Eng. 14, 1 (2001).

12. G. H. Li and Q. Cui, Bipohys. J. 83, 2457 (2002).

13. B. Halle, PNAS 99, 1274 (2002).

14. F. L. Zhang and R. Brüschweiler, Journal of the American Chemical Society 124, 12654 (2002).

15. C. P. Lin, S. W. Huang, Y. L. Lai, S. C. Yen, C. H. Shih, C. H. Lu, C. C. Huang, and J. K. Hwang, Proteins: Structure, Function, and Bioinformatics 72, 929 (2008).

16. K. L. Xia and G. W. Wei, Chaos 24, 013103 (2014).
17. K. L. Xia and G. W. Wei, Physical Review E 88, 062709 (2013).

18. K. L. Xia, K. Opron, and G. W. Wei, Journal of Chemical Physics 139, 194109 (2013).

19. K. Opron, K. L. Xia, and G. W. Wei, Journal of Chemical Physics 140, 234105 (2014).

20. A. G. de Brevern, A. Bornot, P. Craveur, C. Etchebest, and J.-C. Gelly, Nucleic acids research 40, W317 (2012).

21. R. Jing, Y. Wang, Y. Wu, Y. Hua, and X. Dai, J. Theor. Comput. Sci 1, 1000111 (2014).

22. Z. Yuan, T. L. Bailey, and R. D. Teasdale, Proteins: Structure, Function, and Bioinformatics 58, 905 (2005).

23. X.-Y. Pan and H.-B. Shen, Protein & Peptide Letters 16, 1447 (2009).

24. S. Sonavane, A. A. Jaybhaye, and A. G. Jadhav, Bioinformation 9, 134 (2013).

25. P. Radivojac, Protein Science 13, 71 (2004).

26. M. Vihinen, E. Torkkila, and P. Riikonen, Proteins: Structure, Function, and Genetics 19, 141 (1994).

27. I. Guruge, G. Taherzadeh, J. Zhan, Y. Zhou, and Y. Yang, Journal of Computational Chemistry 39, 407 (2017).

28. D. Bramer and G.-W. Wei, The Journal of chemical physics 149, 134107 (2018).

29. S. Altschul, Nucleic Acids Research 25, 3389 (1997).

30. Edelsbrunner, Letscher, and Zomorodian, Discrete & Computational Geometry 28, 511 (2002).

31. A. Zomorodian and G. Carlsson, Discrete & Computational Geometry 33, 249 (2004).

32. R. Ghrist, Bulletin of the American Mathematical Society 45, 61 (2007).

33. A. Hey, S. Tansley, K. M. Tolle, et al., The fourth paradigm: data-intensive scientific discovery, vol. 1 (Microsoft research Redmond, WA, 2009).

This preprint research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstract=3498858
34. Y. C. Lo, S. E. Rensi, W. Torng, and R. B. Altman, Drug discovery today 23, 1538 (2018).

35. J. Bajorath and J. Bajorath, Chemoinformatics and computational chemical biology (Springer, 2011).

36. M. W. Libbrecht and W. S. Noble, Nature Reviews Genetics 16, 321 (2015).

37. Z. Y. Meng, D. V. Anand, Y. P. Lu, J. Wu, and K. L. Xia, arXiv preprint arXiv:1903.02890 (2019).

38. C. Y. Wu, S. Q. Ren, J. Wu, and K. L. Xia, arXiv preprint arXiv:1804.06990 (2018).

39. H. Edelsbrunner, Weighted alpha shapes, vol. 92 (University of Illinois at Urbana-Champaign, Department of Computer Science, 1992).

40. G. Bell, A. Lawson, J. Martin, J. Rudzinski, and C. Smyth, arXiv preprint arXiv:1709.00097 (2017).

41. L. Guibas, D. Morozov, and Q. Mérigot, Discrete & Computational Geometry 49, 22 (2013).

42. M. Buchet, F. Chazal, S. Y. Oudot, and D. R. Sheehy, Computational Geometry 58, 70 (2016).

43. G. Petri, M. Scolamiero, I. Donato, and F. Vaccarino, PloS one 8, e66506 (2013).

44. J. Binchi, E. Merelli, M. Rucco, G. Petri, and F. Vaccarino, Electronic Notes in Theoretical Computer Science 306, 5 (2014).

45. Z. X. Cang and G. W. Wei, PLOS Computational Biology 13, e1005690 (2017).

46. Z. X. Cang, L. Mu, and G. W. Wei, PLoS computational biology 14, e1005929 (2018).

47. R. J. M. Dawson, Cahiers de Topologie et Géométrie Différentielle Catégoriques 31, 229 (1990).
48. S. Q. Ren, C. Y. Wu, and J. Wu, Rocky Mountain Journal of Mathematics 48, 2661 (2018).

49. Z. X. Cang and G. W. Wei, International journal for numerical methods in biomedical engineering p. 10.1002/cnm.2914 (2017).

50. Z. Cang and G.-W. Wei, International Journal for Numerical Methods in Biomedical Engineering 34, e2914 (2017).

51. C. S. Pun, K. Xia, and S. X. Lee, SSRN Electronic Journal (2018).

52. Z. Cang, L. Mu, and G.-W. Wei, PLOS Computational Biology 14, e1005929 (2018).

53. Z. Cang and G.-W. Wei, PLOS Computational Biology 13, e1005690 (2017).

54. A. E. Hoerl, Chemical Engineering Progress 58, 54 (1962).

55. R. Tibshirani, Journal of the Royal Statistical Society: Series B (Methodological) 58, 267 (1996).

56. J. Fan and R. Li, Journal of the American Statistical Association 96, 1348 (2001).

57. H. Zou and T. Hastie, Journal of the Royal Statistical Society: Series B (Statistical Methodology) 67, 301 (2005).

58. H. Zou, Journal of the American Statistical Association 101, 1418 (2006).

59. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Journal of Machine Learning Research 12, 2825 (2011).

60. L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, Classification And Regression Trees (Routledge, 1984).

61. T. K. Ho, in Proceedings of 3rd International Conference on Document Analysis and Recognition (IEEE Comput. Soc. Press, 1995).

62. L. Breiman, Machine Learning 45, 5 (2001).
63. T. Chen and C. Guestrin, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 16* (ACM Press, 2016).

64. J. Bennett and S. Lanning, Proceedings of KDD Cup and Workshop 2007 pp. 3–6 (2007).

65. H. Drucker, C. J. C. Burges, L. Kaufman, A. Smola, and V. Vapnik, in *Proceedings of the 9th International Conference on Neural Information Processing Systems* (MIT Press, 1996), NIPS’96, pp. 155–161.

66. C. Cortes and V. Vapnik, *Machine Learning* **20**, 273 (1995).

67. A. J. Smola and B. Schlkopf, *Statistics and Computing* **14**, 199 (2004).

68. S. Walczak and N. Cerpa, *Information and Software Technology* **41**, 107 (1999).

69. J. Hertz, A. Krogh, R. G. Palmer, and H. Horner, *Physics Today* **44**, 70 (1991).

70. L. R. Medsker, *Hybrid Neural Network and Expert Systems* (Springer US, 1994).

71. E. Barnard and L. F. A. Wessels, *IEEE Control Systems* **12**, 50 (1992).

72. V. Cherkassky and H. Lari-Najafi, *IEEE Expert* **7**, 43 (1992).

73. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, *Journal of Machine Learning Research* **15**, 1929 (2014).

74. F. Chollet et al., *Keras*, [https://keras.io](https://keras.io) (2015).

75. D. E. Tronrud, *Journal of Applied Crystallography* **29**, 100 (1996).

76. D. K. Smith, P. Radivojac, Z. Obradovic, A. K. Dunker, and G. Zhu, *Protein Science* **12**, 1060 (2003).

77. T. G. Project, *GUDHI User and Reference Manual* (GUDHI Editorial Board, 2015), URL [http://gudhi.gforge.inria.fr/doc/latest/](http://gudhi.gforge.inria.fr/doc/latest/).

78. *Dionysus: the persistent homology software*, Software available at [http://www.mrzv.org/software/dionysus](http://www.mrzv.org/software/dionysus).
Figure 1: (a) The 3D model of RNA 4x4u with each chain highlighted in different colour. In this RNA, we are using chain B (highlighted in red). (b) A 2D illustration of a 3D local neighbourhood with a central C1 atom as viewed from a specific axis. Each element is represented by a different colour. (c) Atoms which are within the defined vicinity of the central C1 atom are used for PH. Persistent barcode is generated for C, N, O and P respectively. (d) Topological features are then represented using a binning approach to construct a structured vector form for each element. The four structured vector can be used individually or combined together for ML training.
Figure 2: Effect of Euclidean cut-off distance on RF using the topological features from element P and all four elements with a fixed F/E ratio of 1.0 and bin size of 0.15Å.

Figure 3: Effect of F/E ratio on RF using the topological features from all four elements and bin size of 0.15Å.
Figure 4: Effect of bin size on RF performance with a fixed F/E ratio of 1.0
Table 1: Best test set performance for each ML model. PSSM stands for Position Specific Scoring Matrix.

| Feature type | ML model | Test set PCC | Improvement (%) |
|--------------|----------|--------------|-----------------|
| PSSM         | SVM (RBF)| 0.5028       | -               |
| ESPH - O     | Ridge    | 0.4283       | -14.8%          |
| ESPH - O     | LASSO    | 0.4667       | -7.2%           |
| ESPH - P     | RF       | 0.5788       | 15.1%           |
| ESPH - P     | XGBoost  | 0.5748       | 14.3%           |
| ESPH - P     | SVM (RBF)| 0.5520       | 9.8%            |
| ESPH - P     | ANN      | 0.5732       | 14.0%           |
| ESPH - CNOP  | Ridge    | 0.4849       | -3.6%           |
| ESPH - CNOP  | LASSO    | 0.4157       | -17.3%          |
| ESPH - CNOP  | RF       | 0.5822       | 15.8%           |
| ESPH - CNOP  | XGBoost  | 0.5657       | 12.5%           |
| ESPH - CNOP  | SVM (RBF)| 0.5560       | 10.6%           |
| ESPH - CNOP  | ANN      | 0.5609       | 11.6%           |
| Feature type | ML model | Euclidean cut-off distance | F/E ratio | Bin size | Test set PCC |
|--------------|----------|---------------------------|-----------|----------|--------------|
| ESPH - O     | Ridge    | 25                        | 0.7       | 1.50     | 0.4283       |
| ESPH - O     | LASSO    | 25                        | 0.5       | 0.50     | 0.4667       |
| ESPH - P     | RF       | 45                        | 0.7       | 0.15     | 0.5788       |
| ESPH - P     | XGBoost  | 45                        | 0.9       | 1.00     | 0.5748       |
| ESPH - P     | SVM (RBF)| 40                        | 0.5       | 1.00     | 0.5520       |
| ESPH - P     | ANN      | 45                        | 1.0       | 1.00     | 0.5732       |
| ESPH - CNOP  | Ridge    | 35                        | 0.6       | 0.50     | 0.4849       |
| ESPH - CNOP  | LASSO    | 25                        | 0.5       | 0.50     | 0.4157       |
| ESPH - CNOP  | RF       | 40                        | 0.5       | 0.15     | 0.5822       |
| ESPH - CNOP  | XGBoost  | 45                        | 0.7       | 0.15     | 0.5657       |
| ESPH - CNOP  | SVM (RBF)| 35                        | 0.5       | 0.15     | 0.5560       |
| ESPH - CNOP  | ANN      | 45                        | 0.5       | 0.15     | 0.5609       |
Table 3: Optimal hyperparameters for each ML model

| ML model | Hyperparameters                   | ESPH - $\chi$ | ESPH - CNOP |
|----------|-----------------------------------|---------------|-------------|
| Ridge    | Alpha                             | 500           | 500         |
| LASSO    | Alpha                             | 0.01          | 1           |
| RF       | No of trees                       | 500           | 2000        |
|          | No of min samples at nodes        | 5             | 5           |
| XGBoost  | No of trees                       | 50            | 50          |
|          | Tree depth                        | 3             | 3           |
| SVM      | Kernel RBF                        | RBF           | RBF         |
|          | Gamma 0.01                        | 0.01          | 0.001       |
|          | C 0.1                             | 0.1           | 0.1         |
| ANN      | No of hidden layers               | 4             | 3           |
|          | No of nodes per hidden layer      | 68            | 900         |
|          | Activation type for hidden layer  | Sigmoid       | Sigmoid     |
|          | Dropout rate 20%                  | 20%           | 20%         |
|          | No of epochs 15                   | 15            | 10          |
Table 4: PCC of each RNA chain in training set achieved by the best optimal RF.

| Chain | End-to-end distance | Chain size | Percentage of training set | Chain PCC |
|-------|---------------------|------------|-----------------------------|-----------|
| 1asy_R | 74.52               | 67         | 0.2%                        | 0.9079    |
| 1b23_R | 87.30               | 66         | 0.2%                        | 0.9101    |
| 1c0a_B | 74.72               | 68         | 0.2%                        | 0.9308    |
| 1ddy_A | 56.63               | 35         | 0.1%                        | 0.8727    |
| 1f1t_A | 53.37               | 32         | 0.1%                        | 0.8237    |
| 1ffk_9 | 110.70              | 121        | 0.4%                        | 0.8976    |
| 1ffy_T | 81.21               | 74         | 0.3%                        | 0.8686    |
| 1g1x_D | 83.59               | 39         | 0.1%                        | 0.8760    |
| 1g59_B | 78.86               | 75         | 0.3%                        | 0.9121    |
| 1gax_C | 85.39               | 75         | 0.3%                        | 0.9375    |
| 1h3e_B | 74.16               | 80         | 0.3%                        | 0.8953    |
| 1h4q_T | 72.18               | 64         | 0.2%                        | 0.9117    |
| 1i6u_C | 58.87               | 37         | 0.1%                        | 0.8654    |
| 1j1u_B | 81.41               | 73         | 0.3%                        | 0.8438    |
| 1kxk_A | 95.33               | 69         | 0.2%                        | 0.9157    |
| 1l9a_B | 113.22              | 124        | 0.4%                        | 0.9527    |
| 1m5k_B | 102.67              | 91         | 0.3%                        | 0.9526    |
| 1qf6_B | 73.34               | 69         | 0.2%                        | 0.9550    |
| 1s03_A | 68.57               | 47         | 0.2%                        | 0.9036    |
| 1ser_T | 66.79               | 62         | 0.2%                        | 0.8586    |
| 1sj3_R | 79.10               | 72         | 0.3%                        | 0.9228    |
| Chain   | End-to-end distance | Chain size | Percentage of test dataset | Chain PCC |
|---------|---------------------|------------|----------------------------|-----------|
| 1ttt_D  | 83.16               | 62         | 0.2%                       | 0.8942    |
| 1u9s_A  | 115.07              | 155        | 0.6%                       | 0.9097    |
| 1vfg_C  | 47.42               | 31         | 0.1%                       | 0.7358    |
| 1vy5_AX | 87.82               | 72         | 0.3%                       | 0.8837    |
| 1xjr_A  | 65.92               | 46         | 0.2%                       | 0.8916    |
| 1y26_X  | 70.52               | 70         | 0.2%                       | 0.9126    |
| 1y27_X  | 68.09               | 67         | 0.2%                       | 0.8980    |
| 1yfg_A  | 87.14               | 64         | 0.2%                       | 0.9810    |
| 1yls_B  | 57.26               | 33         | 0.1%                       | 0.7753    |
| 1zho_B  | 49.06               | 38         | 0.1%                       | 0.7856    |
| 2azx_C  | 78.27               | 72         | 0.3%                       | 0.9117    |
| 2cky_A  | 62.84               | 77         | 0.3%                       | 0.8479    |
| 2csx_C  | 77.30               | 75         | 0.3%                       | 0.9007    |
| 2dlc_Y  | 79.89               | 63         | 0.2%                       | 0.7992    |
| 2du3_D  | 77.64               | 71         | 0.3%                       | 0.8841    |
| 2fk6_R  | 58.00               | 51         | 0.2%                       | 0.7857    |
| 2gcs_B  | 101.87              | 122        | 0.4%                       | 0.9190    |
| 2gis_A  | 70.62               | 94         | 0.3%                       | 0.9105    |
| 2hoj_A  | 69.93               | 77         | 0.3%                       | 0.8668    |
| 2hvy_E  | 82.76               | 61         | 0.2%                       | 0.7958    |
| 2nue_C  | 71.75               | 45         | 0.2%                       | 0.8690    |
| 2nz4_P  | 123.18              | 140        | 0.5%                       | 0.9433    |
| 2oeu_A  | 63.02               | 42         | 0.1%                       | 0.8001    |
| 2qbz_X  | 105.64              | 153        | 0.5%                       | 0.9729    |
| Chain   | End-to-end distance | Chain size | Percentage of test dataset | Chain PCC |
|---------|---------------------|------------|----------------------------|-----------|
| 2qex_0  | 223.35              | 2740       | 9.8%                       | 0.9631    |
| 2qus_A  | 87.61               | 68         | 0.2%                       | 0.8835    |
| 2qwy_A  | 64.52               | 52         | 0.2%                       | 0.8904    |
| 2r8s_R  | 116.33              | 158        | 0.6%                       | 0.9177    |
| 2vpl_B  | 64.55               | 48         | 0.2%                       | 0.9015    |
| 2xdb_G  | 63.43               | 34         | 0.1%                       | 0.8177    |
| 2zh2_B  | 61.40               | 34         | 0.1%                       | 0.7996    |
| 2zjr_X  | 219.60              | 2685       | 9.6%                       | 0.9635    |
| 2zjr_Y  | 105.12              | 121        | 0.4%                       | 0.8933    |
| 2zue_B  | 81.15               | 75         | 0.3%                       | 0.8856    |
| 2zzm_B  | 75.64               | 84         | 0.3%                       | 0.8865    |
| 2zzn_C  | 74.81               | 71         | 0.3%                       | 0.9379    |
| 3adb_C  | 86.47               | 92         | 0.3%                       | 0.9345    |
| 3akz_F  | 83.85               | 74         | 0.3%                       | 0.8983    |
| 3am1_B  | 81.67               | 81         | 0.3%                       | 0.8846    |
| 3amt_B  | 84.81               | 78         | 0.3%                       | 0.9104    |
| 3cc2_0  | 222.12              | 2740       | 9.8%                       | 0.9641    |
| 3cul_C  | 81.86               | 91         | 0.3%                       | 0.9310    |
| 3dig_X  | 108.56              | 173        | 0.6%                       | 0.9091    |
| 3e5c_A  | 63.14               | 52         | 0.2%                       | 0.9475    |
| 3egz_B  | 78.94               | 65         | 0.2%                       | 0.9070    |
| 3eph_E  | 76.21               | 69         | 0.2%                       | 0.9306    |
| 3f2q_X  | 77.26               | 107        | 0.4%                       | 0.8954    |
| 3fu2_A  | 42.22               | 31         | 0.1%                       | 0.8248    |
| Chain | End-to-end distance | Chain size | Percentage of test dataset | Chain PCC |
|-------|---------------------|------------|-----------------------------|-----------|
| 3g78_A | 121.02              | 388        | 1.4%                        | 0.8987    |
| 3gca_A | 46.73               | 32         | 0.1%                        | 0.7200    |
| 3gs5_C | 61.00               | 35         | 0.1%                        | 0.7956    |
| 3hlm_C | 85.52               | 135        | 0.5%                        | 0.8988    |
| 3hl2_E | 82.55               | 82         | 0.3%                        | 0.9323    |
| 3iab_R | 75.56               | 46         | 0.2%                        | 0.8980    |
| 3irw_R | 76.11               | 90         | 0.3%                        | 0.8960    |
| 3kfu_K | 87.73               | 71         | 0.3%                        | 0.9002    |
| 3ndb_M | 130.49              | 135        | 0.5%                        | 0.9388    |
| 3nnu_D | 62.64               | 34         | 0.1%                        | 0.9603    |
| 3npn_A | 65.37               | 50         | 0.2%                        | 0.8616    |
| 3ouy_C | 62.07               | 35         | 0.1%                        | 0.8929    |
| 4n0t_B | 70.02               | 65         | 0.2%                        | 0.9465    |
| 4o26_E | 63.51               | 47         | 0.2%                        | 0.8791    |
| 4oji_A | 56.33               | 51         | 0.2%                        | 0.7844    |
| 4oog_D | 60.78               | 34         | 0.1%                        | 0.8030    |
| 4p5j_A | 75.21               | 83         | 0.3%                        | 0.9217    |
| 4p95_A | 102.31              | 188        | 0.7%                        | 0.8799    |
| 4pdb_I | 59.44               | 38         | 0.1%                        | 0.8718    |
| 4pkd_V | 85.57               | 54         | 0.2%                        | 0.9054    |
| 4pqv_A | 62.13               | 68         | 0.2%                        | 0.8610    |
| 4pr6_B | 81.05               | 71         | 0.3%                        | 0.8924    |
| 4qlm_A | 78.00               | 108        | 0.4%                        | 0.8728    |
| 4rdx_C | 79.59               | 76         | 0.3%                        | 0.8612    |
| Chain    | End-to-end distance | Chain size | Percentage of test dataset | Chain PCC |
|----------|---------------------|------------|----------------------------|-----------|
| 4rge_A  | 61.66               | 54         | 0.2%                       | 0.8438    |
| 4ts0_X  | 92.27               | 42         | 0.1%                       | 0.9281    |
| 4u3m_1  | 250.52              | 3149       | 11.2%                      | 0.9667    |
| 4u3m_2  | 265.88              | 1750       | 6.2%                       | 0.9619    |
| 4u3m_3  | 111.81              | 121        | 0.4%                       | 0.9540    |
| 4u3m_4  | 162.52              | 158        | 0.6%                       | 0.9299    |
| 4u7u_L  | 129.24              | 60         | 0.2%                       | 0.9674    |
| 4v51_BA | 218.17              | 2771       | 9.9%                       | 0.9266    |
| 4v67_AA | 243.28              | 1503       | 5.4%                       | 0.9346    |
| 4v67_BB | 106.55              | 118        | 0.4%                       | 0.8578    |
| 4v8b_AB | 91.51               | 87         | 0.3%                       | 0.9551    |
| 4v8d_AB | 90.47               | 84         | 0.3%                       | 0.9329    |
| 4v90_AV | 88.85               | 75         | 0.3%                       | 0.9183    |
| 4v9o_AA | 235.27              | 2854       | 10.2%                      | 0.9562    |
| 4v9o_AB | 105.13              | 118        | 0.4%                       | 0.9214    |
Table 5: PCC of each RNA chain in test set achieved by the best optimal RF.

| Chain | End-to-end distance | Chain size | Percentage of test dataset | Chain PCC |
|-------|---------------------|------------|-----------------------------|-----------|
| 1dk1_B | 74.59 | 56 | 0.9% | 0.0239 |
| 1dul_B | 70.38 | 47 | 0.7% | -0.4194 |
| 1kh6_A | 56.33 | 42 | 0.7% | -0.0116 |
| 2bte_B | 82.27 | 78 | 1.2% | 0.3136 |
| 2oiu_P | 72.68 | 71 | 1.1% | -0.1482 |
| 2ozb_C | 53.11 | 32 | 0.5% | -0.2372 |
| 3hjw_D | 88.36 | 57 | 0.9% | 0.2195 |
| 3owi_A | 98.22 | 86 | 1.3% | 0.4812 |
| 3p22_A | 61.22 | 39 | 0.6% | -0.4328 |
| 3q3z_V | 78.02 | 74 | 1.1% | 0.5131 |
| 3rw6_H | 58.09 | 60 | 0.9% | 0.1577 |
| 3sd3_A | 81.21 | 89 | 1.4% | 0.1356 |
| 3ski_A | 64.25 | 66 | 1.0% | 0.3185 |
| 3suh_X | 89.83 | 100 | 1.6% | 0.1145 |
| 3v7e_C | 93.40 | 125 | 1.9% | 0.1623 |
| 3vjr_B | 58.70 | 36 | 0.6% | -0.3115 |
| 3vrs_A | 61.23 | 51 | 0.8% | 0.5918 |
| 3zgz_B | 78.94 | 81 | 1.3% | 0.2161 |
| 4ato_G | 60.24 | 32 | 0.5% | -0.1138 |
| 4c7o_E | 67.90 | 48 | 0.7% | -0.1835 |
| 4fnj_A | 59.37 | 34 | 0.5% | 0.2358 |
| Chain  | End-to-end distance | Chain size | Percentage of test dataset | Chain PCC |
|--------|---------------------|------------|----------------------------|-----------|
| 4frg.B | 76.24               | 84         | 1.3%                       | -0.2167   |
| 4jf2.A | 78.06               | 76         | 1.2%                       | 0.0851    |
| 4jrc.A | 68.34               | 56         | 0.9%                       | -0.1906   |
| 4k27.U | 81.07               | 55         | 0.9%                       | 0.7721    |
| 4kr6.C | 58.05               | 38         | 0.6%                       | 0.2246    |
| 4kzd.R | 102.80              | 83         | 1.3%                       | -0.1545   |
| 4l81.A | 69.99               | 96         | 1.5%                       | 0.2590    |
| 4lnt_RA| 221.56              | 2881       | 44.8%                      | 0.7864    |
| 4m4o.B | 68.42               | 59         | 0.9%                       | -0.4869   |
| 4v9o_BA| 238.96              | 1533       | 23.8%                      | 0.6667    |
| 4wfl.A | 89.57               | 105        | 1.6%                       | -0.0519   |
| 4x4p.B | 58.08               | 36         | 0.6%                       | 0.0530    |
| 4x4u_B | 59.39               | 31         | 0.5%                       | 0.1802    |

End of Table 5