KYOTO UNIVERSITY

DISSERTATION

Design of Computational Models for Analyzing Graph-Structured Biological Data

グラフ構造を持つ生物情報データに対する
計算モデルのデザイン

Author:
WANG Feiqi

Supervisor:
Prof. AKUTSU Tatsuya

February 8, 2022

Abstract

The computational model, which is an effective solution for complicated problems, has gained remarkable achievements in solving biological issues. Most past computational models designed for solving biological issues utilized general mathematical algorithms or machine learning methods (e.g., neural networks with traditional structures). However, many novel algorithms from informatics can be effectively combined with computational models of biological systems to improve performances. In this study, we proposed two novel computational models to study *pseudoknotted RNA secondary structure comparison* problem and *protein kinase inhibitor interaction sites prediction* problem, respectively.

Biology as one of the most influential nature science nowadays is a complex discipline involving physical structure, chemical composition, function, development, and evolution, where the gene and protein play vital roles in several biological processes as typical issues in bioinformatics area. Genes are foremen that hand out instructions to create life, and the difference of genes determines the species diversity. Besides, proteins involving every process of cell life activities are required components of the creature's body. Thus, it is absolute that both the gene and protein are critical issues in biology.

With the strong development of science and technology in the century, many research fields are hard to further make headway due to the limitation of single-field, thus cross-field study is necessary and important to break the situation. Bioinformatics is in a cross-field of informatics and biology, which applies informatics methods to solve biological problems. Many studies of bioinformatics have effectively taken the biology field forward by applying several informatics methods. In the research, two models are proposed to solve biological issues with complex data structures.

A computational approach is developed to compare RNA secondary structures by topological centroid identification and tree edit distance. In the study, a given graph representing an RNA secondary structure is transformed to a tree rooted at one of the vertices constituting the topological centroid that is identified by removing cycles with PEELING processing for the graph. Then, tree-represented RNA secondary structures are compared by using tree

edit distance and functional gene groups defined by Gene Ontology, where the proposed method showed better clustering results according to their GOs than canonical RNA sequence-based comparison. Furthermore, the combination of the tree edit distance and the sequence edit distance showed a better classification of the pseudoknotted RNA secondary structures.

Then, a machine learning model is designed to analyze the mechanism of protein kinases and inhibitors, which is a graph convolutional neural network (i.e., PISPKI model) attached WL Box module. The WL Box is a novel module that assembles multiple switch weights to effectively compute graph features, which is based on the well-known Weisfeiler-Lehman algorithm. The proposed PISPKI model was evaluated by testing with shuffled datasets and ablation analysis using 11 kinase classes. The accuracy of the PISPKI model varied from 83% to 86%, demonstrating the superior performance compared to two baseline models.

Acknowledgements

I would like to appreciate my supervisor Prof.Akutsu who taught me professional knowledge and always kindly gave me helps.

Then, I also want to say thanks to my parents for supporting me so far and my best friends alongside me regardless of happiness, sadness, or depression. Without them, I cannot get over hard times.

People who have glorious dream make the world a better place.

Lastly, I wish everyone in my Ph.D. life have a future they deserve.

Contents

Chapter 1

Introduction

1 Introduction

1.1 Background

Biology is one of the most influential nature science, which involves many complex disciplines such as physical structure, chemical composition, function, development, and evolution. Designing computational models is a common and effective way to solve biological issues nowadays. The graph as a complicated data structure can carry more information and fits to describe biological data. Thus, to design computational models for analyzing graph-structured biological data, there are two sub-issues: (1) selecting and accurately representing valuable data for biological problems; (2) developing models that can effectively process transformed graph data, where dealing with data of genes and proteins is a critical issue in biology.

Genes, deoxyribonucleic acids (i.e., DNA) or ribonucleic acids (i.e., RNA), are key macromolecules for the continuity of creature life. Due to the stability of deoxyribonucleic acid double helix, DNA molecules do not have complicated plane structures (2D) or stereo structures (3D). However, RNA is not, whose single-stranded structure causes RNA molecules to have special secondary structures and tertiary structures depending on different sequences. Herein, the secondary structure plays a vital role in related technology developments such as nanotechnology and RNA-based computing. Four types of bases (i.e., A, U, G, and C) form to helix, stem-loop, and pseudoknot of RNA secondary structure due to different affinities between two bases. Furthermore, the pseudoknot, which is a double-hairpin structure with an extended quasi-continuous double-helical stem region, as the most complex one in RNA secondary structure motifs has significant meanings in biological processes but is hard to be computed and analyzed because it is irregular.

Therefore, studying with pseudoknotted RNA secondary structures is a challenge.

Besides, the inhibitor is a class of molecules that influences protein activities, and the protein kinase is a phosphotransferase enzyme that catalyzes the transfer of phosphate (PO_4^{3-}) during phosphorylation, where protein kinase inhibitors can block activities of kinases and play vital roles to inhibit the addition of phosphate groups to the target protein by interacting with protein kinase residues via electrostatic forces, hydrogen bonding, and van der Waals forces at specific interaction sites. Phosphorylation of proteins, a central reaction to various biological processes and the regulation of most aspects of cell functions, is a common but complex post-translational modification to modulate cell proliferation, differentiation, and apoptosis. Hence, researching the mechanism of protein kinase inhibitors is a valuable way for understanding phosphorylation and related phenomena.

However, previous studies mostly applied general mathematical algorithms and machine learning methods due to lacking professional knowledge of computer science and informatics. Advanced technologies can be utilized to help them solve the problem and improve effectiveness in various aspects (**Fig.1**). Recently, bioinformatics and computational biology made significant contributions to vaccine and drug developments during the COVID-19 pandemic [1][2][3][4], which further confirms that applying computational models on biological issues can achieve significant progress to science.

1.2 Importance of Cross-field

Compared to the last century, it is increasingly harder to propose a novel study or develop an excellent product in a single field. Most studies are limited by views and technical barriers. Furthermore, the most complex

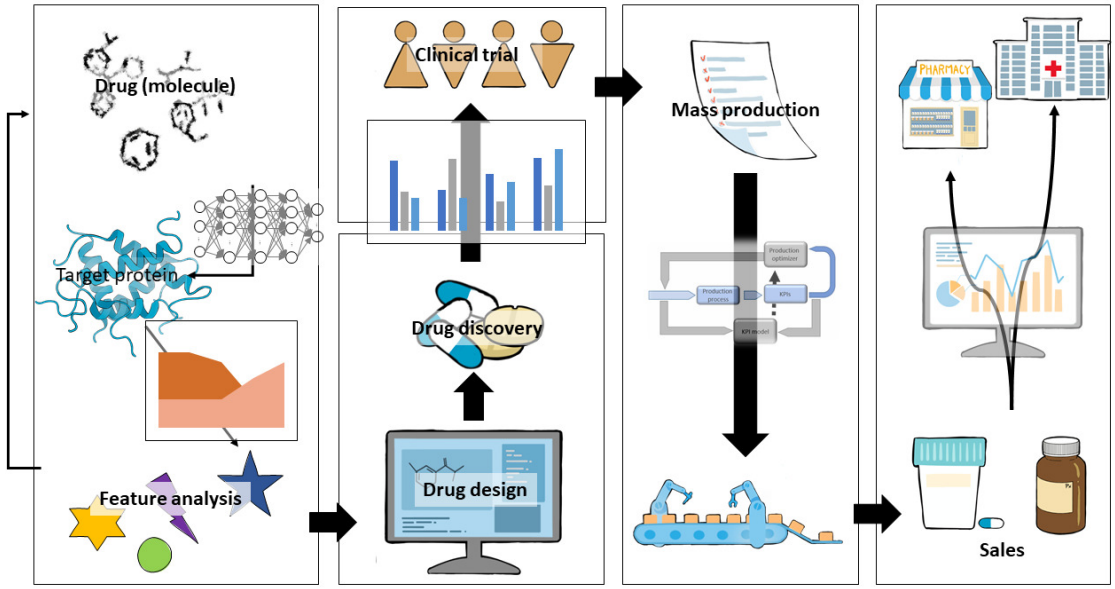


Figure 1: Computational models work on drug discovery procedure

global issues, such as *global warming*, *epidemic*, *cancer*, *food*, *energy*, and *population*, are impossible to be solved by experts from a single field. For example, to solve the *global warming* issue, aerography, agronomy, zoology, bionomics, and informatics are needed. Besides, economic and political issues also should be considered in *global warming*. There are several ways to let different field experts solve a complex issue together, but the communication may not be smooth due to lacking professional knowledge in other fields, which alerts us cross-field experts are needed and cross-field studies should be further progressed.

1.3 Contribution

In this research, we developed two novel methods as shown below.

In the first study, a novel tree representation approach was proposed and

developed based on topological centroid identification and tree edit distance, which was applied to compare and analyze pseudoknotted RNA secondary structures. The pseudo-knotted RNA secondary structure is one of the most crucial substructures for elucidating their individual functions and promoting medical applications. In the proposed method, a given graph representing an RNA secondary structure is transformed to a tree rooted at one of the vertices constituting the topological centroid that is identified by removing cycles with PEELING processing for the graph. Then, tree-represented RNA secondary structure collected from a public database is compared by using the tree edit distance and functional gene groups defined by Gene Ontology (GO). In experiments, the proposed model showed better clustering results according to their GOs than canonical RNA sequence-based comparison.

In the second study, we firstly applied machine learning method on the binding mechanism research of protein kinases and inhibitors, where atoms of protein kinase inhibitors that have interactions with residues from protein kinases was defined as the interaction sites. To study the issue, we proposed and designed a novel machine learning module called the WL Box that is assembled to the PISPKI (Prediction of Interaction Sites of Protein Kinase Inhibitors) model, which is a graph convolutional neural network (GCN) to predict the interaction sites of protein kinase inhibitors. The WL Box is a module based on the well-known Weisfeiler-Lehman algorithm, which assembles multiple switch weights to effectively compute graph structures and output more significant features. The proposed PISPKI model defeated two baseline models and had good performance on shuffled datasets testing. Furthermore, we also designed and performed an ablation experiment to confirm the architecture rationality of the model.

1.4 Organization

In *Chapter 2*, a few basic background knowledge is briefly introduced due to the cross-field, where we hope every reader can understand the research regardless of their areas on computer science, biology, or other majors.

In *Chapter 3*, an approach based on topological centroid identification and tree edit distance is utilized to compare pseudo-knotted RNA secondary structures. Comparison results are shown by the hierarchical clustering analysis with gene ontology.

In *Chapter 4*, a novel machine learning module is designed and attached to the proposed PISPKI model that is applied to analyze the binding mechanism of protein kinases and inhibitors, whose performance is evaluated by baseline experiment, shuffle dataset testing, and ablation study.

In *Chapter 5*, two proposed studies are summarized along with future work.

Chapter 2

Preliminaries

2 Preliminaries

2.1 Data structure

2.1.1 String structure

The string is a primitive data type that represents a sequence of alphabets or symbols, such as $S = s_1 s_2 ... s_n$, where each element from the string can be optionally edited. It is the most common and simple data structure in computer science and has been widely applied in gene analysis [5][6][7]. However, it only can be utilized to describe limited information such as ordered sequences of DNAs or RNAs, because it is unable to further show more complex structure, which leads to the string structure only having poor performance on most complicated biological issues.

String edit distance is an approach that quantifies and measures the degree of dissimilarity between two strings by counting the minimum number of operations with which one string can be transformed into the other, where the Levenshtein distance is the most well known, proposed by Levenshtein in 1965 [8]. There are three operations: insertion, deletion, and substitution. **Fig.2** shows an example for string edit distance of two RNA sequences, where sequence $AUGCCAUAC$ is transformed to sequence $UGCGCUCAC$ by four edit steps: deleting the base A, substituting a base C with base G, substituting a base A with base C, and inserting a base C. During the computation of the string edit distance, setting *weights* can be utilized to get different results and let the program avoid operations that we do not prefer. For example, suppose *weights* of three operations are 1, the string edit distance between two RNA sequences is 4 in **Fig.2**. If we reset the *weight* of a substitution operation to 3 and do not recompute the edit steps, the edit distance is 8. Whereas, the optimal solution of edit steps has been changed because of the

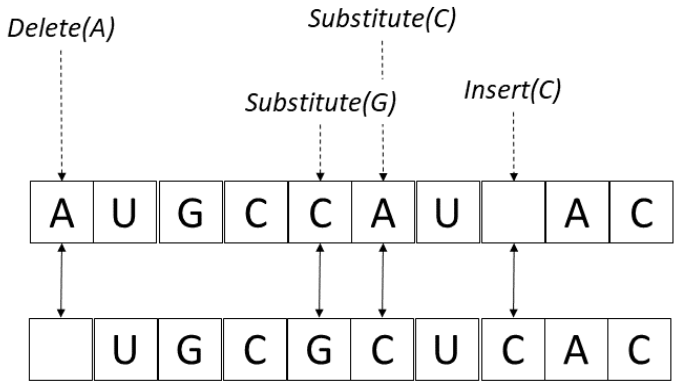


Figure 2: String edit distance between RNA sequence $AUGCCAUAC$ and $UGCGCUCAC$

high *weight* of substitution operations, where each substitution operation should be replaced by one insertion and one deletion operation. Hence, the string edit distance is 6.

2.1.2 Tree

The tree is a data type widely used to describe hierarchical abstract information, which is a finite set consisting of k nodes $n_1, ..., n_k$. There exists a unique node called *root* in a tree T, and the rest of the nodes can be divided into m non-overlapped subsets called subtrees $t_1, ..., t_m$. Subtrees of the tree structure are recursive, which means subordinate subtrees can be continuously found in each subtree. As a hierarchical structure, the tree is acyclic and each node owns a parent except the *root*. The degree of a node represents the number of subtrees, where the node is called *leaf* if and only if the degree equals 0. The *children* of a *parent* node are the nodes that are directly connected with the *parent* node from subtrees. If two nodes have

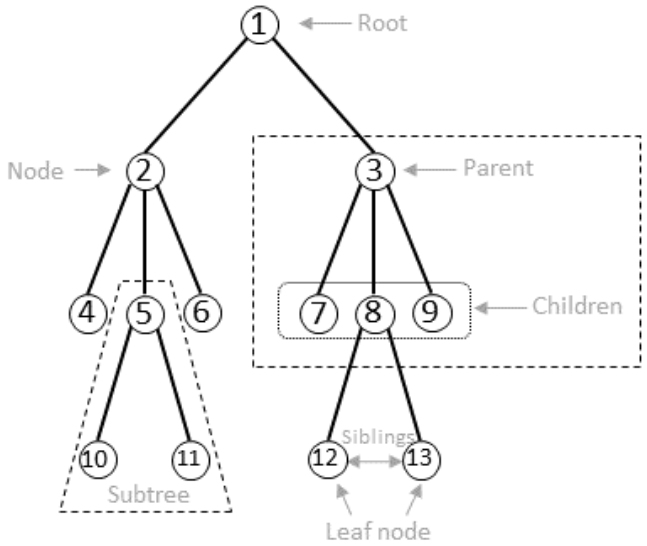


Figure 3: Tree structure

the same *parent*, they are *siblings* (**Fig.3**). Furthermore, The tree can be ordered or unordered. A tree where the children of each node are assigned by specific order is an ordered tree. Otherwise, it is an unordered tree.

The similarity evaluation of two trees can be computed by the tree edit distance. However, complicated structures led to the time complexity of the tree edit distance problem becoming $\mathcal{O}(m^n)$, so optimal solutions of computing minimum edit steps are needed to build. Zhang and Shasha (1989) successfully reduced the time complexity to $\mathcal{O}(m^2n^2)$ by developing an algorithm that applied similar edit operations with the string edit distance (**Section 2.1.1**) [9], where defined i) Insertion: insert a node (*child*) under a node (*parent*) from the tree, where the node can inherit a subtree from the *parent*. ii) Deletion: delete a node from the tree and the *parent* inherit its subtrees if the node is not a *leaf*. iii)Substitution: substitute the label

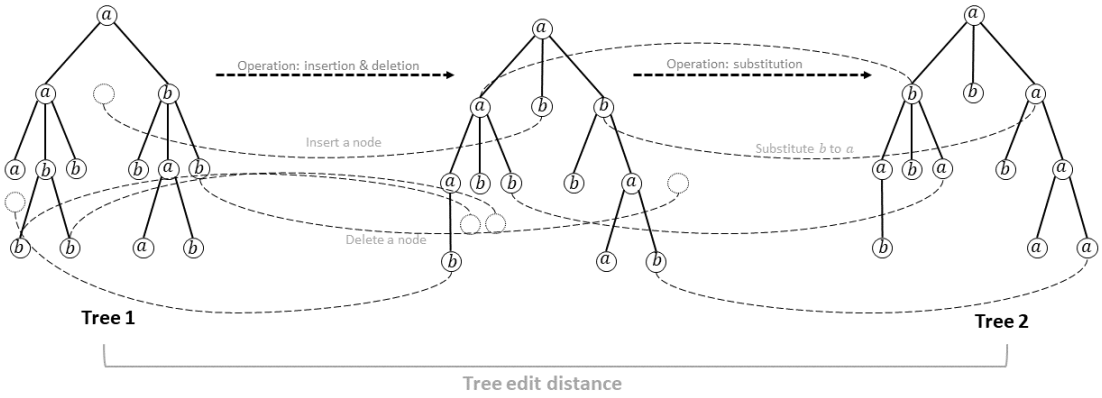


Figure 4: Tree edit distance

of a node from the tree. Similar to the string edit distance, the *weight* can be utilized to avoid unwanted operations (**Fig.4**). Demaine further developed an optimal solution whose time complexity was $\mathcal{O}(n^2m(1+\log(\frac{m}{n})))$ in 2009 [10]. Thus, now, tree edit distance computing is a simple and effective method that calculates the dissimilarity measure of two trees and has been widely applied to computer science issues [11][12][13] but seldom to biology.

2.1.3 Graph theory

The graph is a discrete data structure consisting of a set of n vertices and m edges to describe relationships between vertices by edges represented by $G = (V, E)$, where $V = \{v_1, ..., v_n\}$ and $E = \{e_1, ..., e_m\}$. As shown in **Fig.5**, each vertex can attach information such as *name*, *label*, *color*, and/or *shape* to describe typical targets including proteins or genes in biology. And an edge is formed by two endpoints (vertices) to build the connection between vertices by different relationships, which can be directed or undirected (**Fig.5**). The degree of a vertex denotes the number of edges that are incident to the vertex in a graph denoted by $TD(v)$. Different from the degree of a node from a tree,

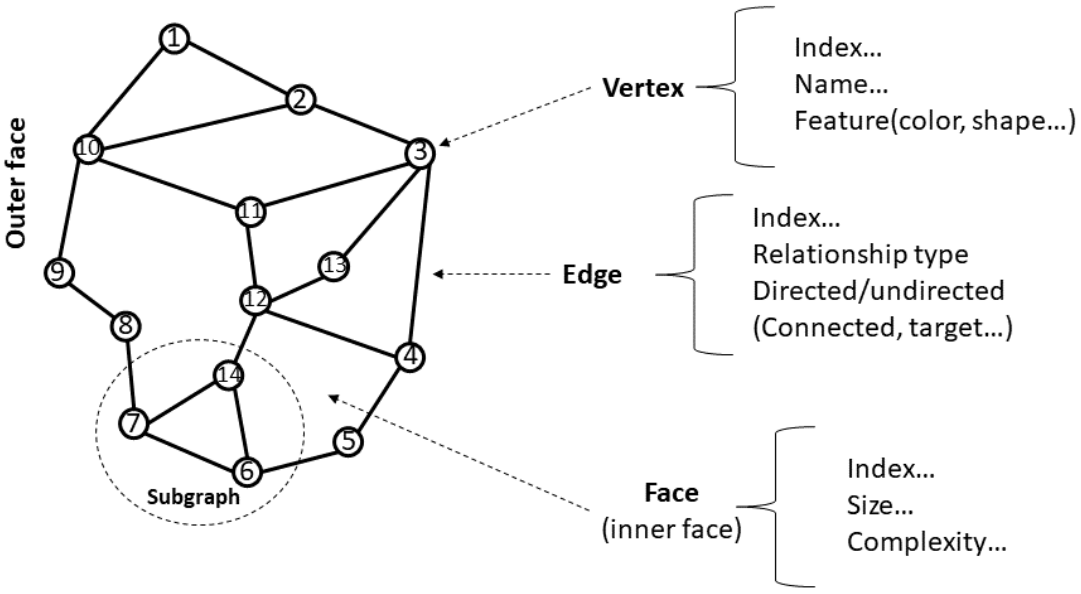


Figure 5: Graph structure

the degree of any vertex cannot be 0 (for connected graphs). For a directed graph, outdegree $OD(v)$ and indegree $ID(v)$ are defined to describe vertice and directions of edges , where $TD(v) = OD(v) + ID(v)$ for any vertices v. Connections of a graph are usually represented by adjacency matrix A that is a square matrix sized by $n \times n$, where A_{ij} is assigned a value 1 if vertices v_i and v_j are connected by an edge. A subgraph of the graph can be built by vertices that are connected with each other, which is formed by a subset of vertices and edges of the graph. The vertex subset must contain all endpoints of the edge subset.

In graph theory, if any nodes from a graph cannot return to itself by only passing each edge once, the graph is an acyclic graph. As mentioned in **Section 2.1.2**, trees are acyclic, which means it is possible to transform acyclic graphs into trees, where different trees can be gotten by selecting different vertices as the *root* (**Fig.6**).

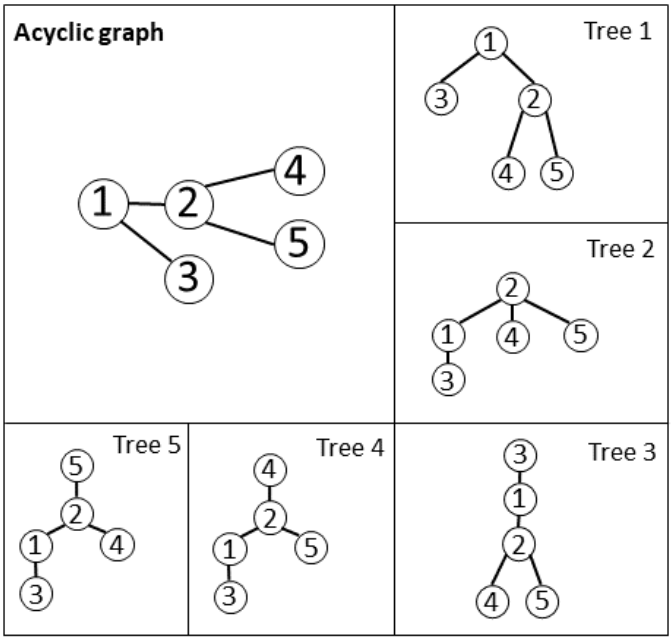


Figure 6: The transformation between acyclic graph and trees

The edit distance can also be utilized to compare two graphs. **Fig.7** shows an example of the graph edit distance between two simple graphs. However, different from the string and tree edit distance, the optimal edit path is needed to be computed in most graph edit distance issues, because the graph does not have a "core" such as the *root* of the tree or the initial element of the string. Hence, bottlenecks still exist in the graph edit distance problem even though neural network approaches have been numerously attempted [14][15][16]. The loss function, which computes the degree of graph deformation, is hard to be defined by edit operations and steps. Furthermore, the time complexity of graph edit distance is exponential ($\mathcal{O}(m^n)$) [17]. Thus, the graph edit distance is not recommended to evaluate complicated graph structures unless efficient algorithms can be developed.

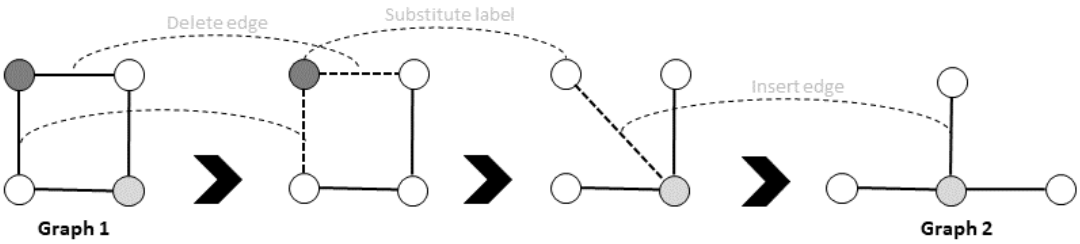


Figure 7: Graph edit distance

2.1.4 Plane graph and Topology

If edges in a graph can be drawn in the plane so that any two edges do not cross, the graph is a planar graph, otherwise it is a non-planar graph. A planar graph embedded in the plane (i.e., a planar graph with a mapping from nodes to points in the plane) is called a plane graph **Fig.8**. Edges from a plane graph split the plane into several regions, which are called face. An inner face of a plane graph is the region bounded by a set of edges where no other vertex or edge exists. The rest of one region that is not the inner face is called the outer face as shown in **Fig.5**. There are three theorems: i) any subgraphs of a plane graph are still plane graphs; ii) if a graph has a non-plane subgraph, the graph must be a non-plane graph; iii) the planarity cannot be changed by adding parallel edges or *loop* [*], which allows limited edits for the structure of a plane graph.

The topology focuses on the invariance of the object in continuous deformations. Let X be a set, and $\mathcal{X}$ be one of the families of sets [†] of X. If the following three conditions hold, the family of sets $\mathcal{X}$ is called a topological space: i)$X, \emptyset \in \mathcal{X}$; ii) if $\alpha, \beta \in \mathcal{X}$, then $\alpha \cap \beta \in \mathcal{X}$; iii) if $\mathcal{X}_1 \subset \mathcal{X}$, then $\cup_{\alpha \in \mathcal{X}_1} \in \mathcal{X}$. In topology, a coffee cup can equal a donut because both of

[*]Loop: an edge that connects a vertex to itself.
[†]Family of sets: a collection of subsets of a given set.

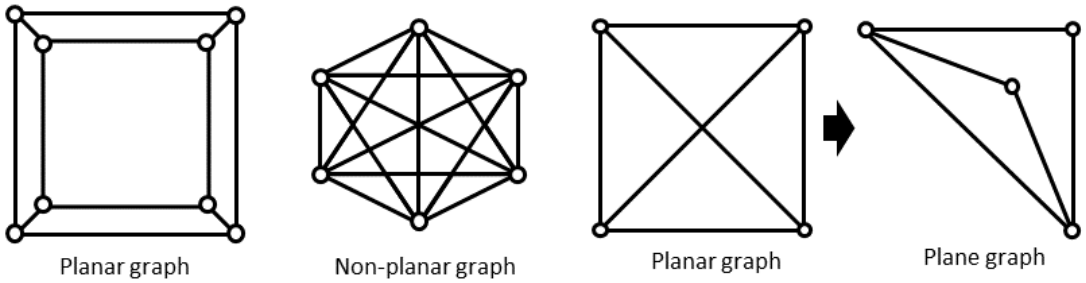


Figure 8: The plane graph

them have one hole, but if the handle of the coffee cup is broken or somebody eats the donut, the topologies of them are changed, which means that the topology concerns more essential properties of objects.

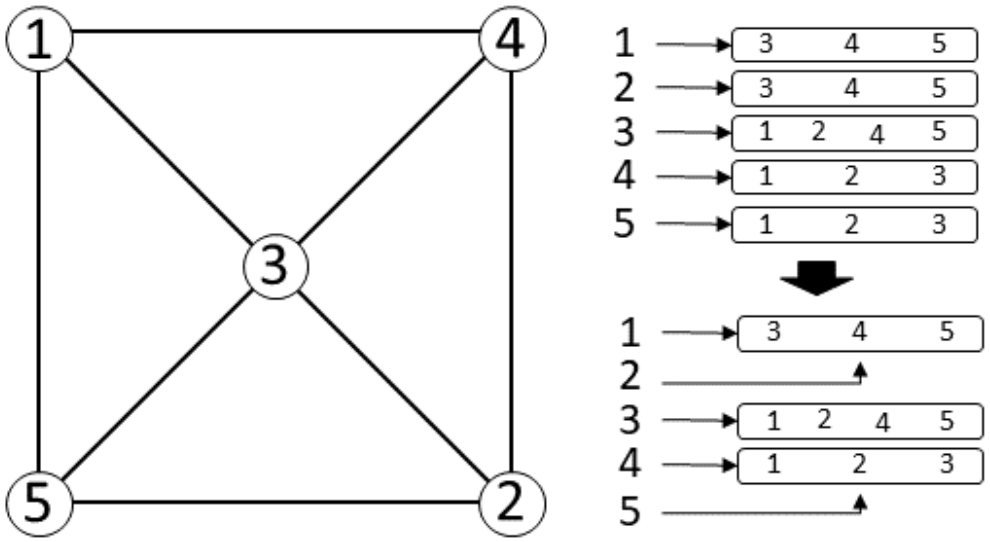


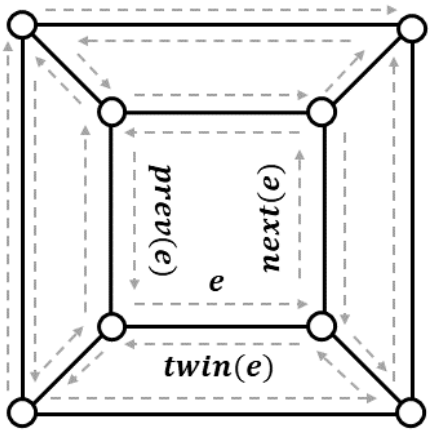
Figure 9: Vertex topology of an example graph

The graph topology is usually applied to optimize network diagrams such as Internet, firewall, router, switch, and LAN, where these successful productions prove the feasibility of applying topology on graph theory. For example, if two vertices have the same connection, the two vertices have the

same topology in the graph (**Fig.9**), then vertices having the same topology can be combined to optimize the graph. Furthermore, more complicated topology operations can be applied to graph theory. Indeed, we combine the topology and data structure transformation approach to solve a biological issue in *Chapter 3*.

2.1.5 Doubly-connected-edge-list

Doubly-connected-edge-list (DCEL) is the most common format that represents an undirected plane graph, which is an edge-based structure including three sets (i.e., vertex, half-edge, and face). Each edge from the plane graph is transformed into two opposite directed half-edges that form a pair of twin half-edges, and half-edges are orderly arranged (i.e., previous half-edge & next half-edge) as shown in **Fig.10**, which facilitates traversing faces and visiting all half-edges around a given vertex to detect the topology of a plane graph.



	Store items
Vertex	*incident half-edge*
Half-edge	*previous half-edge* *next half-edge* *twin half-edge* *target vertex*
Face	*arbitrary half-edge*

Figure 10: Doubly-connected-edge-list structure and data structure

2.2 Artificial Neural network

2.2.1 Traditional neuron

The artificial neural network (ANN) is a bionic application of creatures' nerve cells, which can effectively receive signals and give responses through training and learning. The ANN contains an input layer, an output layer, and several hidden layers. Neurons, which are the most essential components of ANNs, exchange messages with each other. If each neuron of a layer of an ANN exchanges messages with all neurons of the next layer, the ANN is a fully connected neural network. As shown in **Fig.11**, a traditional neuron receives messages from the preceding layer and processes them by weights, activation function, and bias in the fully connected neural network. An output y of a traditional neuron is computed by

$$y = \theta(\sum_i w_i x_i + \beta),$$

where θ denotes an activation function; w_i is a trainable weight; β denotes the bias or threshold; x_i is the input message from ith neuron in the preceding layer.

2.2.2 Graph convolutional neuron

The convolutional neural network is an advanced deep learning architecture based on the traditional neural network (i.e., fully connected neural network), which applies the special convolutional neuron that can accurately process graphs or figures. **Fig.12** shows the procedure of the convolutional neuron. Compared to the traditional neuron, the convolutional neuron can process messages in high dimensions such as matrices, where the trainable multi-dimensional *kernal* is utilized to extract features. The kernel moves from

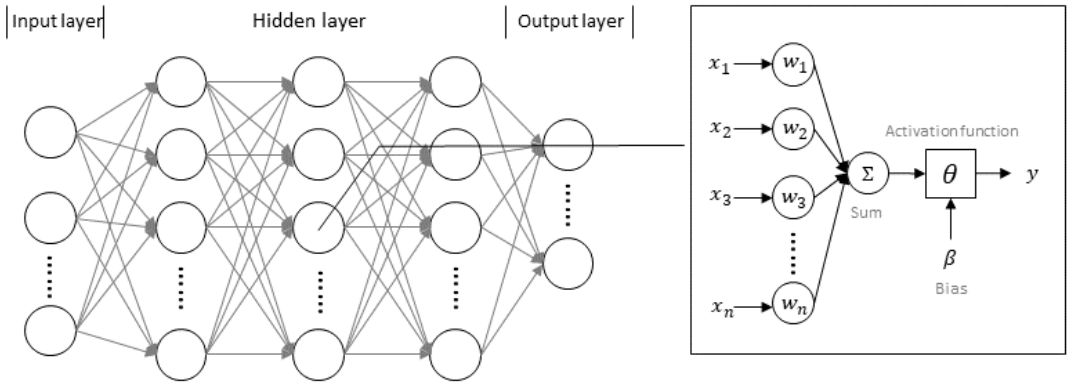


Figure 11: Artifical neural network and tranditional neuron

the top left and the distance of movement depends on the *step*. After being processed by a convolutional neuron, the input matrix is resized to

$$[\frac{W - K_w + P}{S} + 1, \frac{H - K_h + P}{S} + 1],$$

where W and H are the width and height of the input matrix, respectively; the size of convolutional *kernel* K is $[F_w, F_h]$; P and S denote the padding length and *step* in the convolutional neuron.

The output matrix o is computed by

$$o[x, y] = \sum_{i=1}^{K_w} \sum_{j=1}^{K_h} w_{ij} \cdot M[S(x-1) + i, S(y-1) + j],$$

where w_{ij} is the trainable weight at ith row jth column in the kernel K.

In *Chapter 4*, a novel module (i.e., convolutional neuron) is developed based on a well-known *graph isomorphism* algorithm.

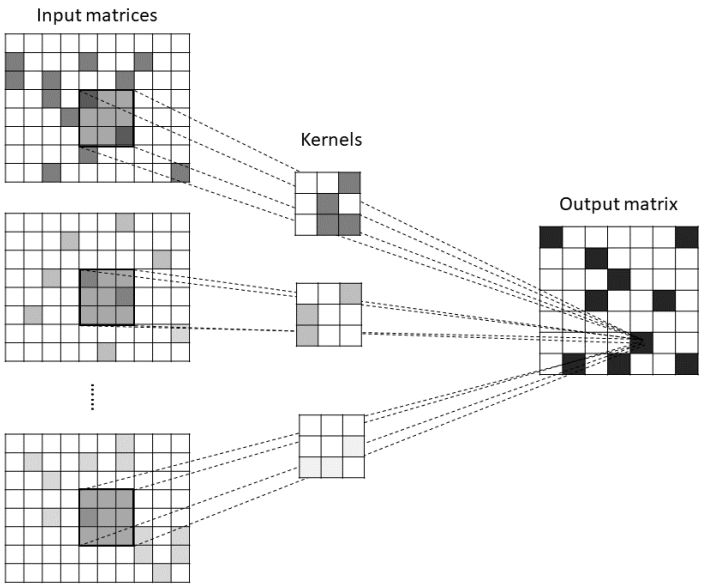


Figure 12: Computation of convolutional neuron

2.2.3 Pooling

Amounts of datasets that are processed by convolutional neural networks are large, where the amounts may further increase after passing through convolutional neurons. Usually, fully connected layers are the last few layers of a convolutional neural network. These large amounts of datasets overload fully connected layers. Hence, pooling is set between the convolutional layer and the fully connected layer, which summarises pooling regions of features by *max*, *average*, or *etc.*, where, the invariance including translation, rotation, and scale of data is further improved. Furthermore, pooling can effectively reduce information redundancy and avoid over-fitting.

Each unit $[x, y]$ of the average pooling output can be computed by

$$p[x, y] = \sum_{i=1}^{K_w} \sum_{j=1}^{K_h} p_0[K_w(x-1)+i, K_h(y-1)+j]/K_w K_h$$

where K_w and K_h denote the width and height of pooling.

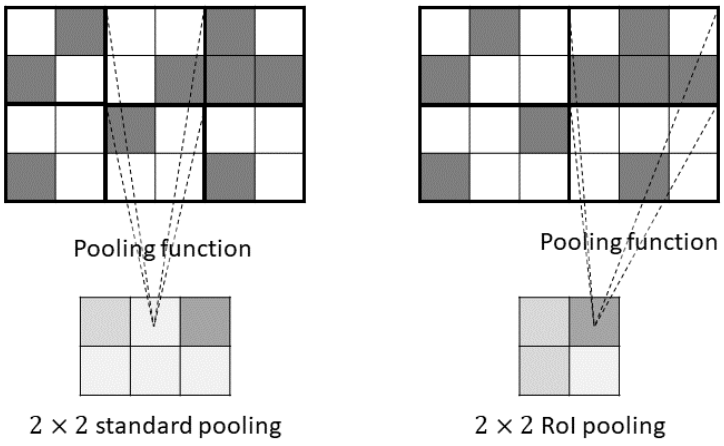


Figure 13: Comparison of strandard pooling and Region of Interest pooling

However, the output size of the standard pooling as introduced above depends on the input from convolutional layers. As mentioned in **Section 2.2.2**, the output size of the convolutional neuron is influenced by kernel size. Besides, the input of the convolutional neural network is unfixed-size in most studies, which could further lead the output of convolutional neurons uncertain. However, messages outputted by pooling are processed by fully connected layers, which need fixed-size inputs. Zero-padding [‡] is commonly utilized to fix the problem, but it may reduce the performance of the model. Region of interest (RoI) pooling can process the input to the fixed-size output. **Fig.13** shows the comparison of standard pooling and RoI pooling.

Each unit $[x, y]$ of average RoI pooling out can be computed by

$$p[x,y] = K_w K_h \sum_{i=1}^{\frac{W}{K_w}} \sum_{j=1}^{\frac{H}{K_h}} p_0[\frac{W}{K_w}\dot{(}x-1)+i, \frac{H}{K_h}\dot{(}y-1)+j]/WH$$

[‡]Zero padding is a technique typically employed to make the size of the input sequence equal to the maximum size of all inputs by padding 0.

where W and H denote the width and height of the input matrix; K_w and K_h represent the size of RoI.

To fit the novel machine learning module developed in this research, spatial pyramid pooling, which is an advanced existing RoI pooling module [18], is applied to the model in *Chapter 4*.

Chapter 3

An Approach for Comparison of Pseudoknotted RNA Secondary Structures

3 An Approach for Comparison of Pseudo-knotted RNA Secondary Structures

3.1 Background

3.1.1 RNA secondary structure

RNAs are an essential component of various biological processes such as the transmission of genetic information from DNAs to proteins and regulation of expression, regulation, coding, decoding of genes. These are polymer macromolecules constituted by nucleic acids with four nucleobases (i.e., Adenines, Uracils, Guanines, and Cytosines). In RNA molecules, nucleotides are basic building blocks of nucleic acids connected by 3'5' phosphodiester bonds to create backbones, which show end-to-end connections of 3' - hydroxyl groups and 5' - phosphate groups between two nucleotides of nucleic acids. The sequence of nucleobases of nucleic acids is the primary structure of the RNA. Different from the double-stranded helix of the DNA, the RNA does not have the two strands connected by hydrogen bonds but has a single-stranded folded onto itself (**Fig.14**). Different affinities between two nucleobases make RNAs have special secondary structures, where Adenines (A) prefer to pair with Uracils (U) and Guanines (G) prefer to pair with Cytosines (C).

It is widely believed and accepted that there is a strong correlation between RNA function and structure, so that prediction and comparison of RNA structures are fundamental analyses to elucidate the functions of individual RNAs. However, the prediction problem of RNA tertiary (3D) structure directly from its sequence is difficult in terms of both biological and computational experiments. Therefore, RNA secondary structures that can be mathematically well-formulated have attracted much attention, and var-

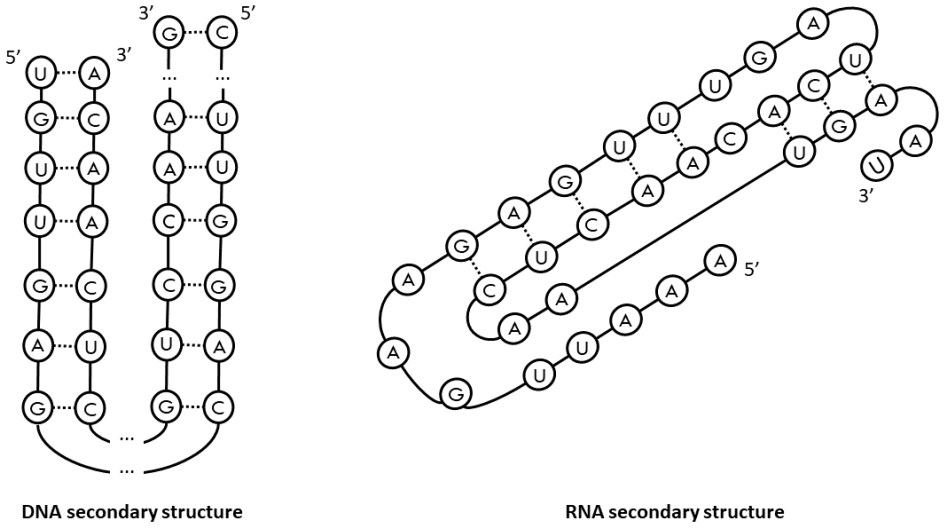


Figure 14: Difference of secondary structure between DNA and RNA

ious algorithms for RNA secondary structure analysis have been proposed. Hofacker et al. developed a software tool called the Vienna RNA package to compute and compare RNA secondary structures [19]. They applied the tool to long RNA molecules, the heat capacity of RNA secondary structures, and RNA hybridization. Dulucq and Tichit transformed RNA secondary structures into tree structures and compared them with focusing on computational complexity based on tree edit distance [20].

3.1.2 Pseudo-knotted RNA

The pseudo-knot is a structural motif in the RNA and one of the most complex secondary structures, which contains at least two stem-loop structures where half of one stem is intercalated between the two halves of the other one. Stems are referred to continuously paired nucleobases (i.e., two complementary sequences), and there are several type stem-loop structures: i)

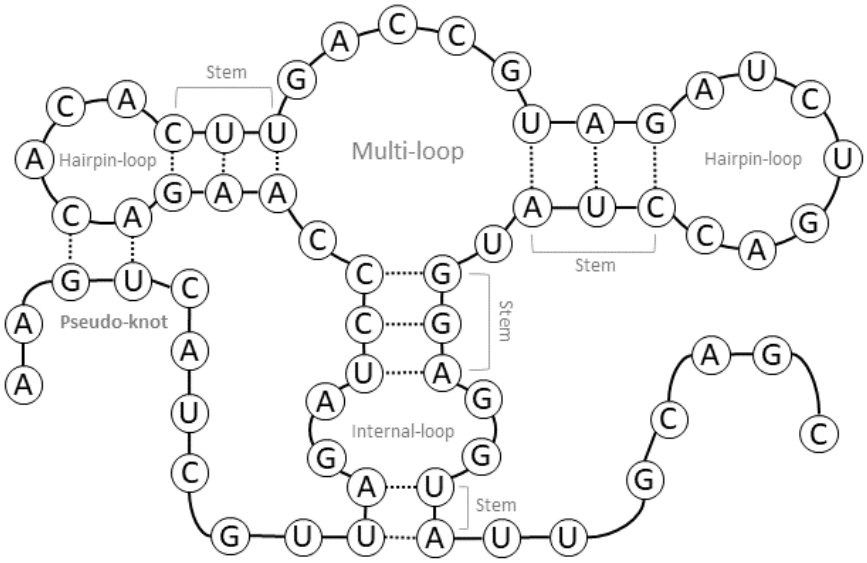


Figure 15: RNA secondary structure

the hairpin-loop consists of a stem joined by non-pairing bases in a loop, ii)the internal-loop includes two stems joined by non-pairing bases separated by one or two loops, iii) the multi-loop is made up by three or more stems joined by non-pairing bases in two or more loops. As shown in **Fig.15**, most RNA secondary structures such as multi-loop or hairpin-loop are regular, but the pseudo-knot may exist in any two stem-loop structures. Pseudo-knots are hard to be detected by existing computational biology approaches such as dynamic programming, which is the reason why the pseudo-knot is known as the most complex RNA secondary structure.

As mentioned above, most methods do not support pseudo-knotted struc-

tures in RNA secondary structure due to their complex prevalent folding motif [21]. Although the pseudoknot has significant meanings in biological processes [22], it is known that both prediction and alignment of pseudoknotted RNA secondary structure are NP-hard problems [23, 24, 25], so that certain pseudoknot structures are often ignored or transformed into pseudoknot-free structures by removal operations for efficient computation [26]. Andronescu et al. developed a constraint-based parameter estimation algorithm to predict RNA secondary structure using pseudoknot-removed data [27]. Moreover, Smit et al. classified RNA structure into structural components for explaining universal compositional patterns in rRNA secondary structure categories, and they discarded pseudoknot and gave up analyzing this region [28]. However, recently, fast and accurate integer programming-based prediction methods for pseudoknotted RNA secondary structures were developed and reported their usefulness in terms of practicality in [29] and [30]. On the other hand, transforming RNA secondary structure to the tree is a common preprocess for the comparison. However, pseudoknotted RNAs are difficult to be reduced into trees due to their structural complexity. Thus there are few methods dealing with the comparison of arbitrary pseudoknotted RNA secondary structures [31]. There is a software package named *RNAforester* that can compare RNA secondary structures but only for pseudoknot-free RNAs [32]. Evans developed a polynomial-time algorithm by restricting the pseudoknot topology to be aligned [33]. In contrast, Möhl et al. aligned arbitrary pseudoknotted RNA secondary structures under an assumption on costs of RNA secondary structure edit operations but the time complexity for the worst-case is exponential [34]. Although PSMAlign succeeded in efficient alignments of pseudonotted structures by identifying similar stem structures, it is a local alignment approach and the worst-case time complexity is also

exponential [35]. From the above, dealing with pseudoknot regions is a big obstacle for RNA secondary structure studies.

3.1.3 Data structure transformation

Data structure transformation is a common approach that simplifies complicated issues. In this thesis, a novel method for comparing pseudoknotted RNA secondary structures is developed based on the tree edit distance via identifying *topological centroids* of input plane graphs representing RNA secondary structures based on *PEELING* algorithm devised in [36] and building *topological centroid trees* from the plane graphs. To show the effectiveness of the proposed method, we perform hierarchical clustering analysis of real pseudoknotted RNA secondary structures obtained from a public database PseudoBase++ [37]. The results suggest that the proposed method classifies the pseudoknotted structures according to their functional groups defined by Gene Ontology (GO) [38, 39] much better than the simple comparison of their sequences based on the string edit distance. In addition, we report a case that a distance metric combining the string edit distance and the tree edit distance also yields better clustering performance for the pseudoknotted structures. The proposed method for transforming a plane graph to a tree, *planeGraph2tree*, is available at (https://github.com/feiqiwang/planeGraph2tree).

3.2 Method

An RNA secondary structure can be described as a plane graph even it is with pseudo-knots [43]. The proposed method (**Fig.16**)firstly identifies the topological centroid of an input plane graph of RNA secondary structure, by **PEELING** algorithm with detecting and removing of specific groups of edges, faces, and subgraphs called *singly exposed edges*, *singly exposed face*,

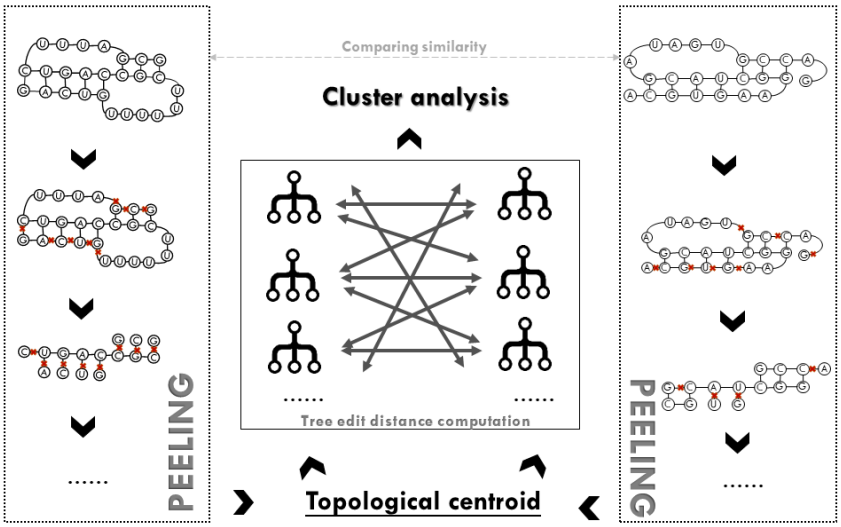


Figure 16: The approach for analyzing pseudo-knotted RNA secondary structures

and *adjunct subgraphs*, respectively [36]. Then, the disjointed plane graph is transformed into a tree (i.e., an acyclic graph) according to *reorganization* rules (**Section3.2.3**). The gotten tree generates a set containing several patterns depending on the type of topological centroid. Based on the above, two pattern sets from two different plane graphs of RNA secondary structures are computed by tree edit distance to analyze the similarity.

3.2.1 PEELING

Let $G(V, E)$ be an undirected graph, where $V = \{v_1, ..., v_N\}$ is a finite set of vertices and E is a finite set of edges. Each vertex is assigned a label $l(v_i) \in \{A, U, G, C, X, Y\}$. The edge is represented by $(v_i, v_j) \in E$ where v_i and v_j are connected in G. A graph is called a *plane graph*, if G is connected and its planar embedding is given. Each edge (v_i, v_j) has a face set $\mathcal{F}_{(v_i,v_j)}$,

where (v_i, v_j) is related to faces in the set. The areas inside and outside of the connected graph are called inner faces and outer face f_{outer} [§], respectively. Let $\mathcal{C}$ be the directed cycle consisting of the edges of the outer face, where edges are visited in the clockwise order as follow:

$$\mathcal{C} = \{(v_i, v_j) | f_{outer} \in \mathcal{F}_{(v_i,v_j)}\}.$$

A function $\Phi((v_i, v_j), (v_j, v_k))$ is defined to determine whether two edges are connected or not, where $\Phi((v_i, v_j), (v_j, v_k)) = 1$ if edges are connected, otherwise, 0. An inner face of G is called *singly exposed* if its outer edges in $\mathcal{C}$ are connected in $\mathcal{C}$ (ignoring direction of edges), where an edge is called an outer one if it also belongs to the outer face. Those edges are collected to a collection that is added to the set

$$\mathcal{P}_f = \{\{(v_i, v_j), ..., (v_l, v_k) | (\mathcal{F}_{(v_i,v_j)} \cap ... \cap \mathcal{F}_{(v_l,v_k)}) - f_{outer} \neq \emptyset$$
$$\wedge \Phi((v_i, v_j), ...) \cdot ... \cdot \Phi(..., (v_l, v_k)) = 1 \wedge f_{outer} \in \mathcal{F}_{(v_i,v_j)} \wedge, ..., \wedge \mathcal{F}_{(v_l,v_k)}\}\}.$$

Similarly, an edge is called *singly exposed* if it is not belonging to an inner face and one of its endpoints is of degree 1. $\mathcal{D}(v_i)$ is a function counting the number of vertex v_i existing in E. Each identified single exposed edge is collected to the set

$$\mathcal{P}_e = \{(v_i, v_j) | f_{outer} \in \mathcal{F}_{(v_i,v_j)} \wedge (\mathcal{D}(v_i) = 1 \vee \mathcal{D}(v_j) = 1)\}.$$

In addition, each maximally connected subgraph surrounded by a singly exposed face with sharing only one outer vertex is called an *adjunct subgraph*, and all edges belong to the subgraph are collected as a list added to the

[§]More detailed defination is shown in **Section2.1.4 Plane graph and topology**

adjunct subgraph set $\mathcal{P}_g$.

3.2.2 Topological centroid identification

The topological centroid is a unique structural feature of the graph, which is generated from topology rules. By ultilizing topological centroid, meaningless information loss can be avoided due to reorganizations of graphs. To identify the topological centroid of a given plane graph G, the **PEELING** repeats the following two procedures until there does not exist any singly exposed face or edge in G: i) identify all singly exposed faces, single exposed edges, and adjunct subgraph, then ii) delete outer edges and adjunct subgraphs in these exposed faces and edges **Fig.17**, where the topological centroid is either a vertex, an edge, or a face.

3.2.3 Reorganization

Topological centroid tree is a tree having its topological centroid identified by **PEELING** as the root. If the topological centroid is either an edge or a face, one of the vertices constituting them is determined as the root. As a result, multiple topological centroid trees are generated from one input plane graph (**Fig.18**).

As mentioned in **Section 3.2.1**, three sets $\mathcal{P}_f$, $\mathcal{P}_e$, and $\mathcal{P}_g$ containing all edges of inputted plane graph except the topological centroid are gotten in **PEELING** procedure. During the building procedure of the topological centroid tree, the acyclic graph is firstly reorganized from the inputted plane graph with three sets following rules: i) the second vertex of the last edge of each list belonging to a single exposed face is replaced by a new generated vertex (leaf node) that is assigned the same label with original vertex such as editing (v_1, v_2) to $(v_1, v_{2'})$ meanwhile $l(v_2) = l(v_{2'})$, ii) single exposed edges

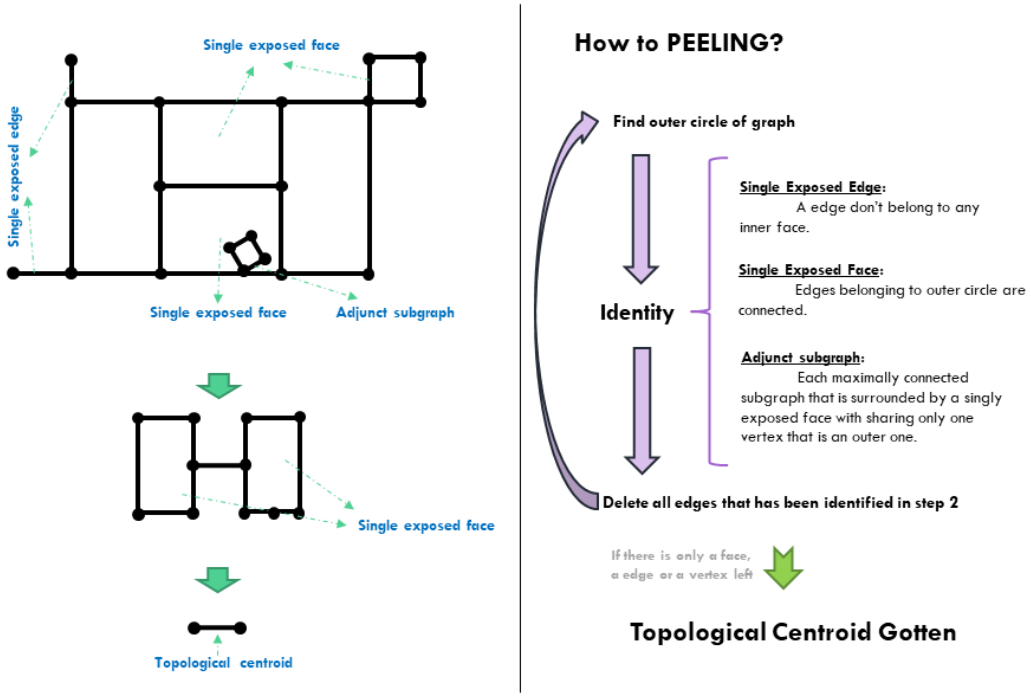


Figure 17: PEELING procedure

in $\mathcal{P}_e$ recover connection relationship with the rest of parts of the graph, and iii) each adjunct subgraph in $\mathcal{P}_g$ is processed by **PEELING** and reorganized itself following i) and ii) then reconnected to the graph.

3.2.4 Tree edit distance

After transforming input plane graphs to topological centroid trees by **PEELING** and reorganization procedure, gotten trees are compared by *tree edit distance* under the unit cost model [41]. The tree edit distance is one of the most widely used measures for comparing tree structured data and it is defined by the cost of the minimum cost sequence of edit operations for

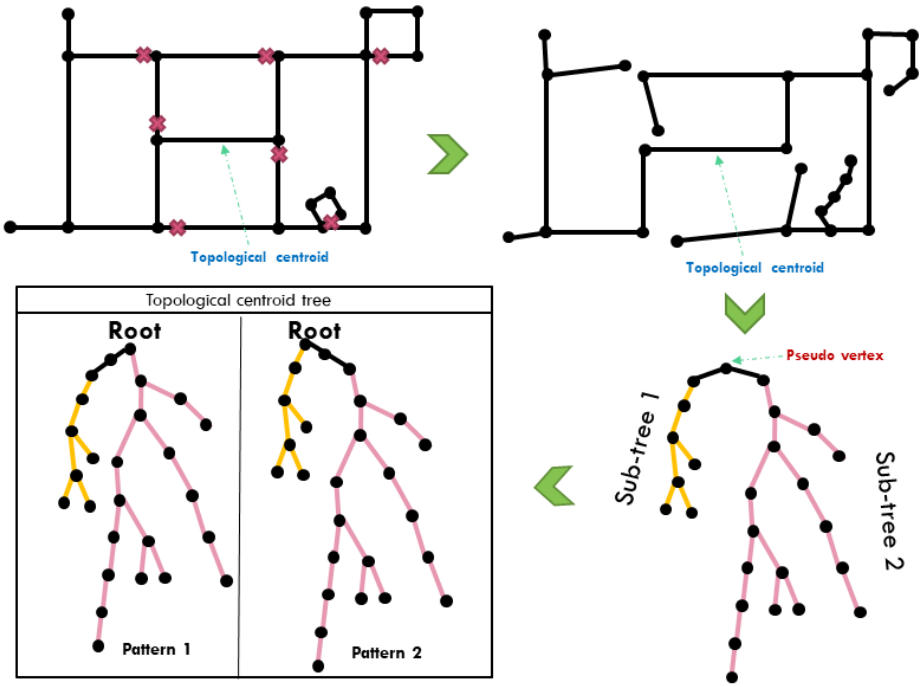


Figure 18: Building procedure of topological centroid tree

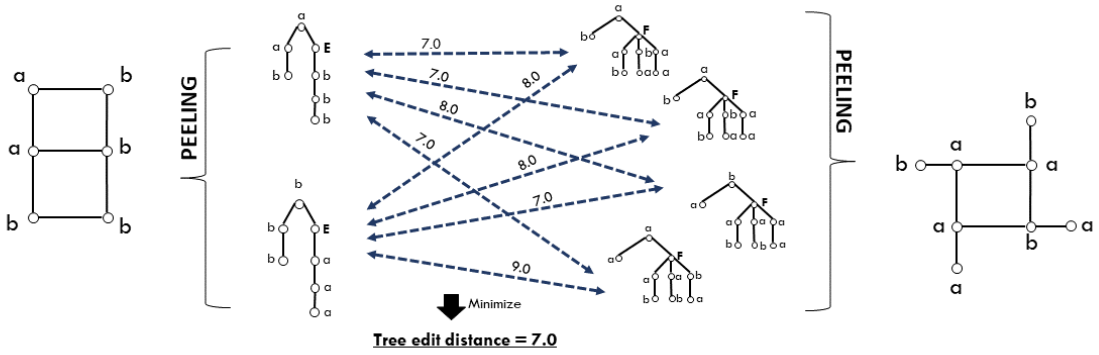


Figure 19: Computation of the topological centroid tree edit distance

transforming a tree into another tree, where an edit operation is either a deletion of a non-root node, an insertion of a non-root node, or a substitution of the label of a node. The unit cost model means that the cost of each edit operation is 1. Although the tree edit distance is computed between a pair of trees, as explained in the last section, there are multiple possibilities for which vertex becomes root in each topological centroid tree when its topological centroid is either an edge or a face. Therefore, here we define the *topological centroid tree edit distance* as the minimum tree edit distance for all combinatorial pairs of the trees with different root nodes derived from respective input topological centroid trees (**Fig.19**).

3.3 Experiment

To demonstrate the effectiveness, the proposed method was applied to a real dataset of pseudoknotted RNA secondary structures. Since it is widely accepted that there is a strong correlation between the structures and functions of biological molecules, we perform the hierarchical clustering analysis that classifies the pseudoknotted structures based on the similarity computed by

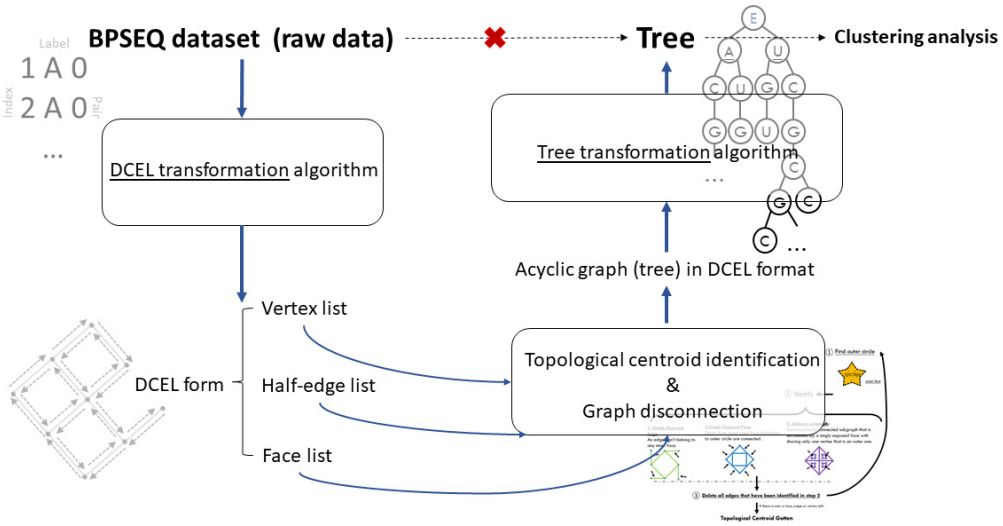


Figure 20: Experiment procedure

the (topological centroid) tree edit distance and the functional groups defined by GO. The experiment procedure is shown in **Fig.20**. In this study, experimental datasets were collected in BPSEQ format, then an algorithm (DCEL converter) was developed for transforming *BPSEQ* dataset to the doubly-connected-edge-list (DCEL) format, which supplied suitable input datasets to planeGraph2tree program that was developed based on **PEELING** (**Section.3.2.1**) and reorganization rules (**Section.3.2.3**). Costs (*weights*) of tree edit distance operations (i.e., inseration, deletion, and substitution) are uniformly set to 1.

3.3.1 Database

304 pseudoknotted RNA secondary structures were collected from PseudoBase++ [37]. The dataset is orderly annotated by PKB-number (PKB1–PKB304) and each of them is described in *BPSEQ* format, which simply

represents data structure by three columns; i) the first column is the index ordered from 5' to 3' for an RNA sequence, ii) the second column contains the label that acronym of nucleotide type which is either of A, U, G, C, X, or Y, where X and Y means the nucleotide would be any type and C or U, respectively, and iii) the third column informs the index of paired partner for the nucleotide, and it is set to be 0 if the nucleotide is unpaired.

On the other hand, the **PEELING** algorithm assumes that an input plane graph is given by the double-connected-edge-list (DCEL) format [36]. The DCEL is an edge-based data structure for easily manipulating and traversing a plane graph, and it contains the three sets of record (i.e. half-edges, vertices, and faces) to represent topology of a plane graph. Each edge on a plane graph is represented by two opposite directed half-edges called twins, and all half-edges should be linked by their heads and tails each other. Each half-edge stored an incident vertex, an incident face, and next/previous half-edges while each vertex and face stored an incident half-edge, respectively.

Thus, when applying **PEELING**, the collected dataset was transformed from BPSEQ format to the DCEL format as a preprocessing. However, the detailed information of PKB71 and PKB75 are not provided in the database, so that the 302 pseudoknotted structures except for them were selected from the 304 structures.

3.3.2 DCEL converter

A C++ program (i.e., DCEL converter) was developed to transform *BPSEQ* into DCEL format. The algorithm is shown in **Appendix Algorithm 2**. i) All nucleobases are represented as vertices and connected by two inverse half-edges, where a vertex stores the label of nucleobases and the incident

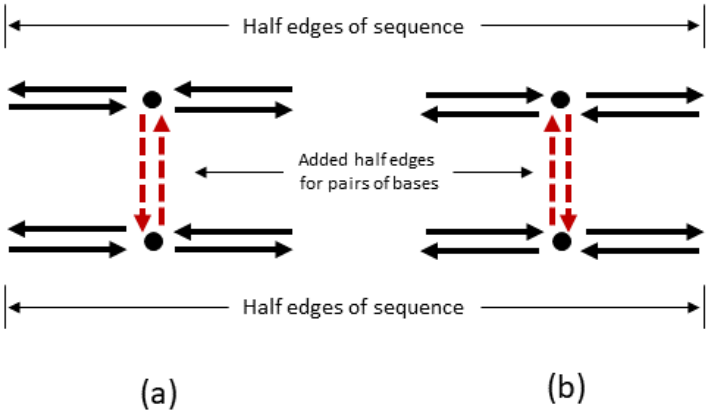


Figure 21: Two ways to add half-edges

half-edge. And each half-edge stores its previous half-edge, next half-edge, twin half-edge, and target vertex (*line 1-12*). ii) The pair information from the *BPSEQ* text guide program connecting two vertices (nucleobases), where some store items of related half-edges are rewritten. There are two ways to add half-edges as shown in **Fig.21**. A direction-swap rule is built to swap the way to add half-edges. The program continuously works in one way (**(a)** or **(b)** in **Fig.21**) unless the direction-swap rule holds (*line 13-40*). iii) Each loop of half-edges is a face storing the index and an arbitrary half-edge from the loop, which is recorded in set F. (*line 41*) In the end, the DCEL plane graph was identified by Euler's formula (i.e., $n(V) - n(E) + n(F) = 2$[¶]), where a developed repair sub-program will run if the formula is not tenable (*line 42*).

3.3.3 Gene ontology

Gene ontology (GO) is a major representation method in bioinformatics to unify the gene attribute, where three GO domains (i.e., biological process,

[¶]n(*) is a function that counts the number of the set and n(E) equals half of the number of half-edges.

molecular function, and cellular component) are utilized to describe the functions of genes. The biological process (BP) domain refers to operations of molecular events that involve complicated biological processes in creature bodies such as DNA repair or signal transducer [42]. The molecular function domain describes the molecular level of elemental activities with products such as transport [44]. The cellular component is about the structural situation of a gene product in a cell [45]. Since three domains describe different features and gene products play different roles in biological processes, a gene can have multiple gene ontology terms even in the same domain. Each GO term has unity and unique identification such as GO:0006401 that can be indexed in the Gene Ontology Resource database which contains the domain, aspect, definition, relationship of a GO. In each domain, there is a directed acyclic graph (tree) to describe the relationship between GO terms. The GO term [child node] is a part of the GO term [parent node]. **Appendix Fig.S1** shows an acyclic relationship graph between nine GO terms, which is quite complicated. This study aims to compare and analyze RNA (gene) secondary structures, but evaluation could be complicated if genes have too many GO terms. Thus, focusing on genes with fewer GO terms should be recommended in this study.

3.3.4 Algorithm validation evaluation

For the collected 302 pseudoknotted RNA secondary structures, GO information was obtained through EMBL number [46] and Rfam database [47]. However, 100 out of the 302 structures were removed from this analysis since their EMBL number or Rfam information was lacking. As for the remaining 202 structures, 111 structures were further extracted to which only one GO term was assigned in order to simplify our clustering analysis, and

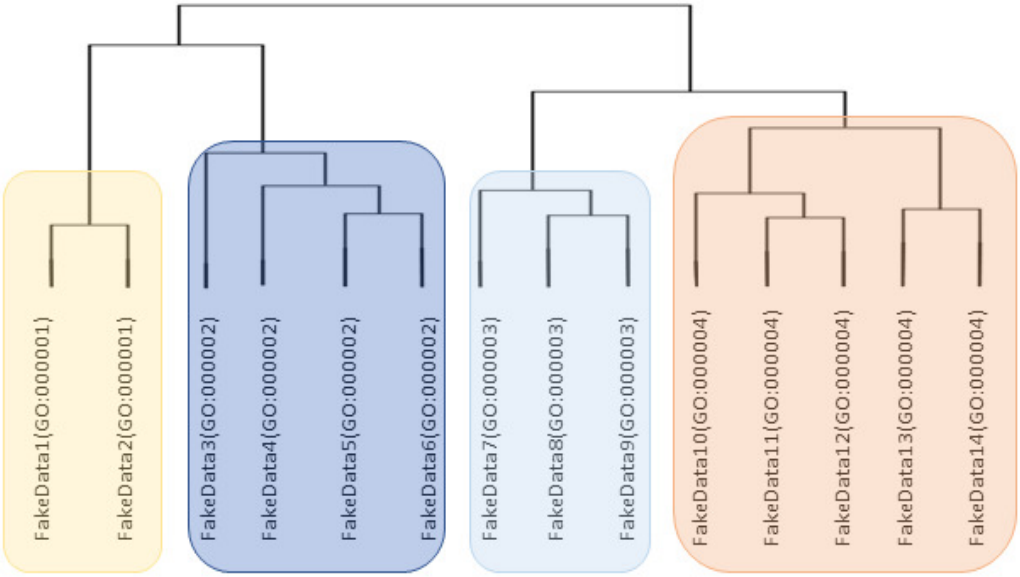


Figure 22: An example of ideal clustering result based on gene ontology terms

they include thirteen different GOs (GO:0000372, GO:0006401, GO:0006412, GO:0006417, GO:0006452, GO:0019079, GO:0039689, GO:0039705, GO:0043022, GO:0045069, GO:0045727, GO:0046782, and GO:0075523). As mentioned in **Section 3.3.3**, GO terms describe typical features for genes. Thus, if our approach can effectively identify RNA secondary structures that have the same or similar GO terms, the validation can be proved. The hierarchical clustering analysis is applied and **Fig.22** shows an example of an ideal clustering result based on gene ontology terms.

3.4 Results

79 pseudonotted structures assigned either one of the four GOs (GO:0043022 GO:0046782 GO:0006417, and GO:0075523) which were most frequently included in the 111 structures were extracted. The hierarchical clustering re-

sults for the 79 structures by *hclust* with *ward* method on a programming language R are shown in **Fig.23**. The input distance matrix for the *hclust* was computed by RTED, which is a software for computing the tree edit distance between ordered trees [48]. The labels show PKB numbers and GO numbers, and **Fig.23(a)** and **(b)** correspond to the results of the string edit distance-based comparison of the RNA sequences and our tree edit distance-based comparison of the RNA secondary structures, respectively. As shown in the results, the RNAs were classified better according to their functions by the tree edit distance-based RNA comparison than the sequence-based comparison. For example, the RNAs with GO:0075523 were classified to one branch in the tree edit distance-based comparison whereas they were separated into two branches in the sequence-based comparison. The tendency can also be seen in GO:0006417, that is, most of the RNAs with GO:0006417 except for three RNAs (PKB119, PKB120, and PKB180) are classified into one branch in the tree edit distance-based comparison but those were divided into two branches in the sequence-based comparison.

To further evaluate the effectiveness of our comparison approach, we demonstrated another clustering analysis for a larger set of GOs and classified them according to their two super families, macromolecule metabolic process (MMP) and viral process (VP). For the 13 GOs from the 111 structures selected in the last paragraph, GO:0043022 is classified to the category of molecular function while rest of 12 GOs are classified to the category of biological process. Therefore, we selected 77 pseudoknotted RNA secondary structures of the 12 GOs with removing the 34 structures of GO:0043022 in the following analysis. Among the 12 GOs, the six and the five GOs have the same ancestor MMP (GO:0043170) and VP (GO:0016032), respectively, and GO:0039705 is a common offspring of MMP and VP (**Fig.24**).

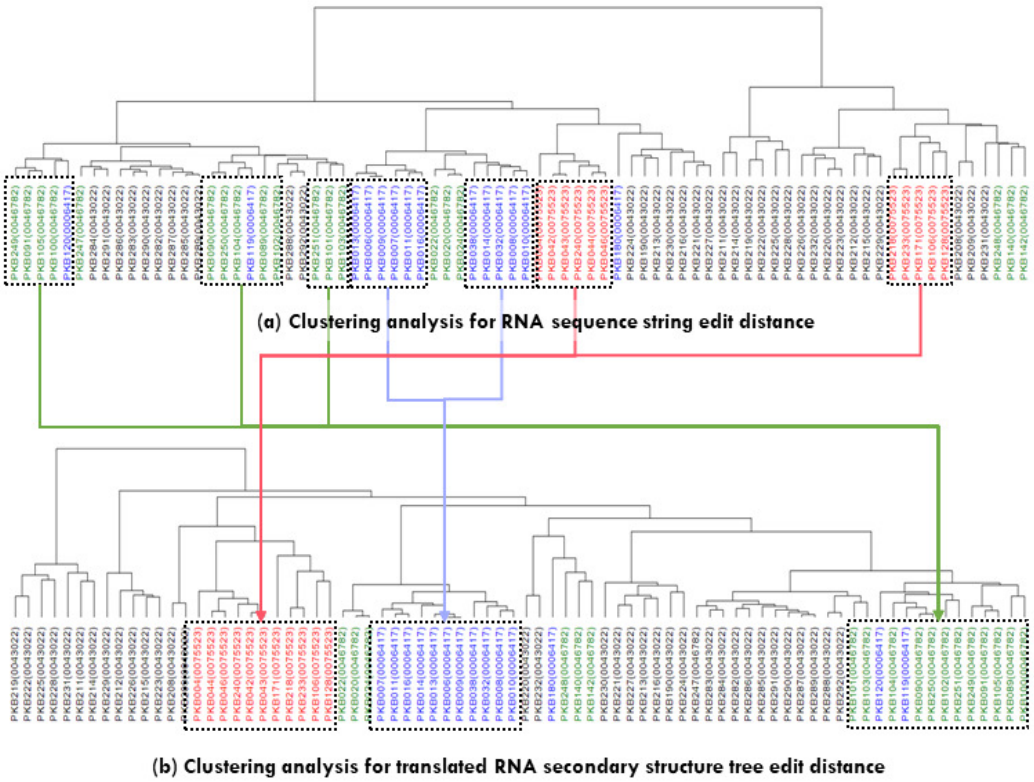


Figure 23: Hierarchical clustering results for pseudoknotted RNA secondary structures based on four GOs

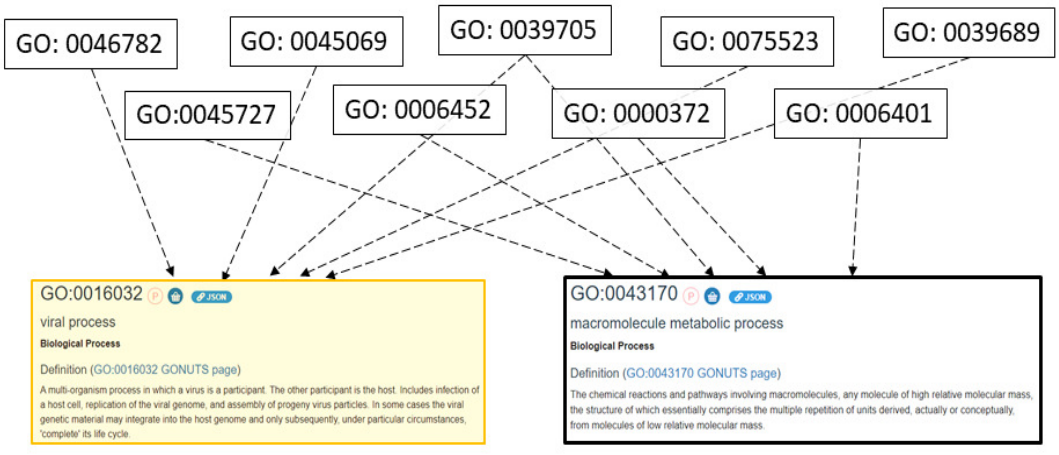


Figure 24: GO functional group classification

The clustering results of the 77 RNA structures with 12 GOs from the two super families of MMP and VP are shown in **Fig.25**. **Fig.25(a)** and **(b)** are the clustering results based on the string edit distance and the tree edit distance for the structures, respectively. In this experiment, the clustering result of a combination of the string edit distance and the tree edit distance is also shown in **Fig.25(c)**, where the combination means that the distance metric is defined by the sum of the edit distance and the tree edit distance. As a result, there was no significant difference between the clustering results based on the string edit distance and the tree edit distance. However, 10 structures of MMP separated into two or three different branches in these two methods were clustered in one branch in the combination method (See labels surrounded by red boxes in **Fig.25**). Since they have similar secondary structures, it can be concluded that there exits a case that the sensitivity for the difference in pseudoknotted RNA secondary structures transformed from the plane graphs was increased by combining the string edit distance and the tree edit distance.

3.5 Discussions

A novel approach for comparing pseudoknotted RNA secondary structures based on the tree edit distance by transforming input plane graphs representing RNA secondary structures to trees via the topological centroid computing was introduced in this study. In the computational experiments using real pseudoknotted RNA secondary structures collected from a public database PseudoBase++, the proposed method showed much better hierarchical clustering results of the structures according to their GOs than the simple sequence-based comparison. In addition, the effectiveness of using a distance metric combining the string edit distance and the tree edit distance

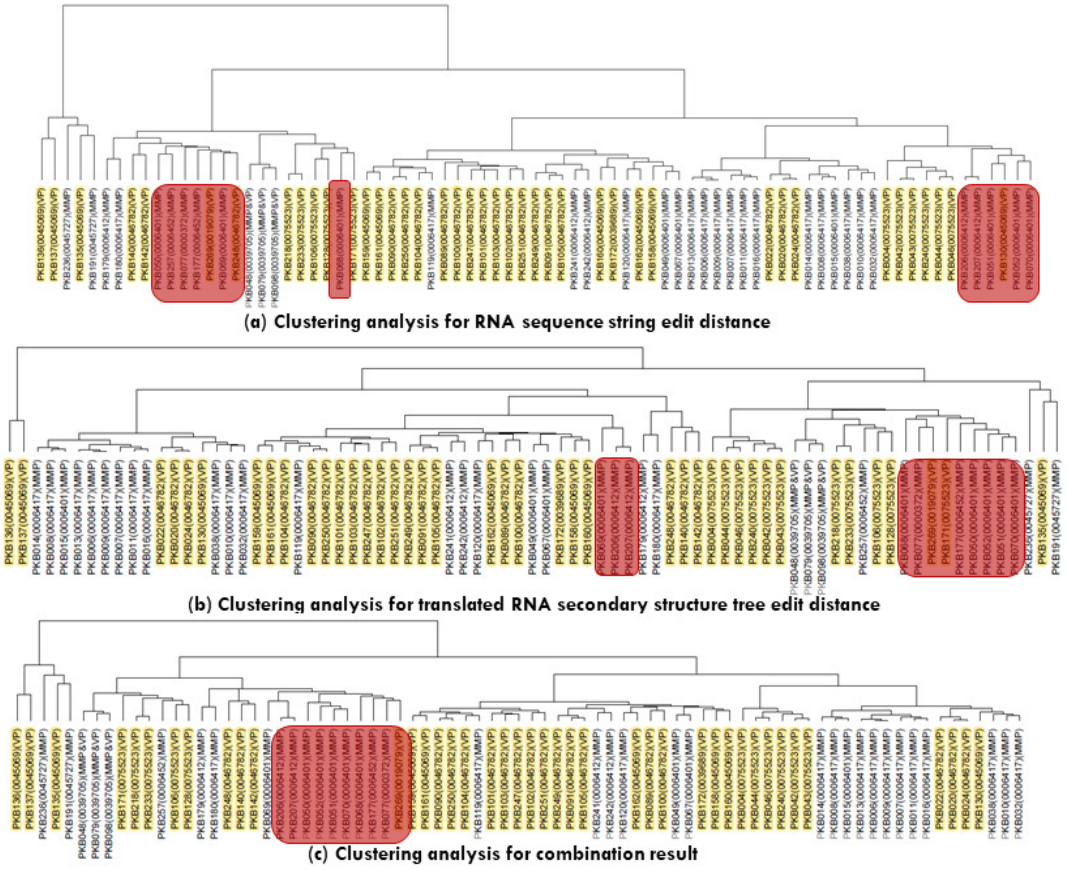


Figure 25: Hierarchical clustering results for pseudoknotted RNA secondary structure based on larger functional groups MMP and VP

was also shown.

However, the secondary structural information was degenerated when converting a plane graph into a tree, so that a complete comparison has not been achieved. For example, when focusing on a certain branch of a tree, whether it was originally a single exposed edge or a resulting edge of **PEELING** for a single exposed face is not taken into account in the comparison. In addition, when there exist a single exposed edge and a single exposed face having the same label sequence, it may be difficult to distinguish them since they become identical each other after **PEELING**.

There are other points to note besides the loss of the structural information. In this comparison method, a tree structure is given as an ordered tree, that is, an order relationship from left to right is defined between child vertices. Indeed, an RNA secondary structure is often represented as an ordered tree, but a tree generated by the proposed method has no order relationship between vertices disconnected by **PEELING**. Thus, it is considered to be appropriate to treat it as an unordered tree. The tree edit distance computation between unordered trees, however, is known to be NP-hard [49], and there is no effective algorithm that can compute the distance in polynomial time. Therefore, in this study, a tree is treated as an ordered tree in order to efficiently compute the tree edit distance while sacrificing some accuracy.

Finally, of note, we collected pseudoknotted RNA secondary structures from PseudoBase++ and performed clustering analysis according to GO information based on the assumption that there exits a strong correlation between the structures and functions of biological molecules. However, PseudoBase++ provides not the whole structure but the partial structure for each RNA, so that it may be insufficient to evaluate the clustering results in terms of their functions. Nevertheless, the fact that the proposed method showed

better clustering results than the simple sequence comparisons suggests that the partial structures of RNAs may contribute to their functions.

Chapter 4

A Novel Prediction Model for Interaction Sites of Protein Kinase Inhibitors

4 A Novel Prediction Model for Interaction Sites of Protein Kinase Inhibitors

4.1 Background

Phosphorylation of proteins, which is central to various biological processes and the regulation of most aspects of cell functions [50], is a common but complex post-translational modification to modulate cell proliferation [51], differentiation [52], and apoptosis [53]. Many studies about protein post-translational modification have effectively taken the biology field forward by using machine learning methods [54][55].

4.1.1 Protein kinase in phosphorylation

A protein kinase is a phosphotransferase enzyme that catalyzes the transfer of phosphate (PO_4^{3-}) groups donated by high-energy adenosine triphosphate (ATP) molecules to specific residues in order to regulate activities of proteins [56][57][58][59][60][61]. Because phosphorylation is an important biochemical process, protein kinases have been investigated as potential therapeutic targets [62][63][64][65][66]. In addition, kinase inhibitors block the activities of kinases and are vital to inhibit the addition of phosphate groups to the target protein [67]. Here, exploring the binding mechanism plays a crucial role on kinase inhibitor design. Many studies on the development of molecular drugs have focused on protein kinase inhibitors for the treatment of infectious diseases [68] and cancers [69].

4.1.2 Inhibitor

During the process of protein phosphorylation, the γ-phosphate group of the ATP molecule is replaced by a hydroxide ion from water that is hydrolyzed to an inorganic phosphate ion existing in the environment [70]. Afterward, protein kinases transport the inorganic phosphate ions to the residues of the protein substrates [71], which are typically serine, threonine, or tyrosine residues [72]. Based on the specific phosphorylated residue, these molecules are classified as serine/threonine, tyrosine-specific, histidine-specific, and aspartyl/glutamyl protein kinases [73]. Although there exist various classes of protein kinases, the characteristics of members of the same class are homologous [74][75]. However, protein kinases can be incorporated into protein-ligand complexes that bind to molecular inhibitors [76] that block the transportation process [77]. Kinase inhibitors interact with protein kinase residues via electrostatic forces, hydrogen bonding, and van der Waals forces at specific interaction sites. We define those atoms that have interactions with residues from protein kinases as the interaction sites for inhibitors in this research.

4.1.3 Contribution

Bioinformatics is a versatile tool to research complicated biological processes, and machine learning continues to gain popularity for the development of tools to analyze biological data. A graph convolutional neural (GCN) network is a recently developed neural network to directly operate and analyze graphic structures and has been widely applied for analysis of protein-ligand complexes, structure-embedded graph representation [78], structure-based virtual screening [79], prediction of binding affinity [80][81], and prediction

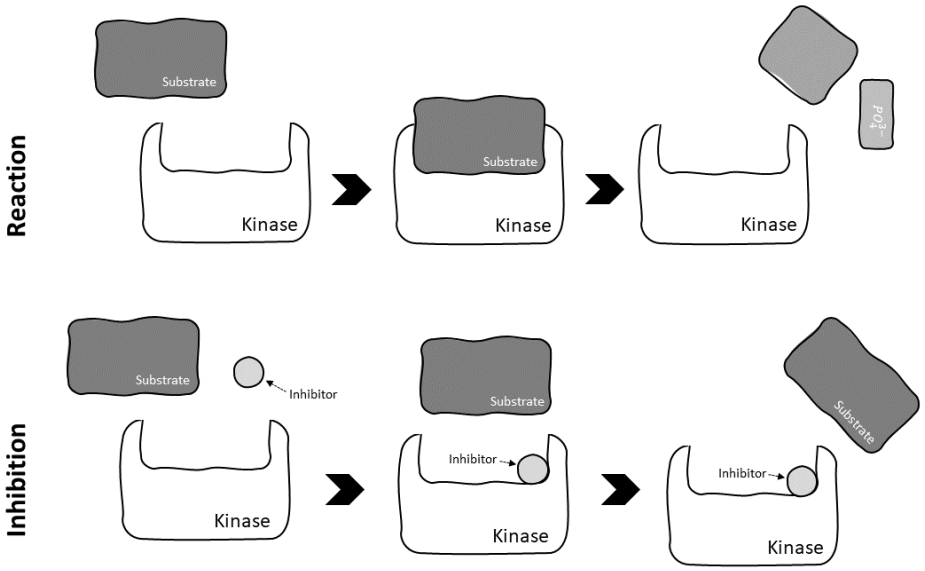


Figure 26: The inhibition process for protein kinase

of binding residues [83]. Moreover, many novel algorithms have been proposed for solving specific biological issues in recent years [82]. Although most previous studies have focused on issues with protein-ligand complexes, to our knowledge, there is no previous study on the prediction of the interaction sites of inhibitor molecules based on known protein kinase-ligand complexes. As compared with affinity prediction, the prediction of interaction sites with a GCN network is more intuitive, allowing for the collection of the features of protein kinase inhibitors for designing more effective drug design. Here, a novel machine learning module, the Weisfeiler-Lehman (WL) Box, was designed and a GCN network with WL Boxes was developed as a tool to predict the interaction sites of different classes of protein kinase inhibitors. The WL Box is based on an algorithm proposed in 1968 by Weisfeiler and Lehman to solve the *graph isomorphism problem* [84]. Most past studies that utilized the Weisfeiler-Lehman algorithm on machine learning modules simply pro-

cessed vertex and edge information as inputs. However, we flexibly applied edge information to develop novel components (i.e., *switch weights*) on proposed modules. To the best of our knowledge, this is the first application of a GCN network to predict the interaction sites of protein kinase inhibitors. The result confirmed that the WL Box is an effective tool for the analysis of protein kinase inhibitors and drug prediction studies.

4.2 Method

4.2.1 Preliminaries

Here, an inhibitor molecule is defined as an undirected graph denoted by G, which is represented by a 2-tuple (F, S), where F is a feature matrix representing the feature of the vertices and S is an adjacency matrix representing relationships among the vertices for N atoms and N_e type bonds. Set $C = \{c^1, ..., c^m, ..., c^{N_c}\}$ consists of N_c color types, where each color c^m is represented by an N_c-dimensional binary vector. For each color vector $c^m = (c_1^m, c_2^m, ..., c_i^m, ..., c_{N_c}^m)$, element c_i^m is assigned a value of 1 if and only if $m = i$; otherwise the element is assigned a value 0.

Let $G(V, E)$ be an undirected graph representing an inhibitor molecule, where $V = \{v_1, \ldots, v_N\}$ is a set of atoms and E is a set of edges. Information on atoms (i.e., vertices in G) is represented by a binary feature matrix F of size $N \times N_c$ in which each row corresponds to an atom and the corresponding row vector is a color vector representing the atom type. Information on the edges of G is represented by an adjacency matrix S of size $N \times N$. S_{ij} (i.e., the element of the ith row and jth column) is assigned a value of 0 if $\{v_i, v_j\} \notin E$, otherwise matrix S_{ij} denotes the bond type (i.e., $S_{ij} \in \{1, 2, \ldots, N_e\}$).

Since graphs representing chemical structures are also considered, there is no self-loop; thus all diagonal elements of S are assigned a value 0. To effectively utilize the adjacency matrix, each vertex v_i is assigned a label index l_i according to the feature matrix row color c^m by

$$l_i = m.$$

Then, the convolutional layer input matrix S_{conv} is obtained by the structure adjacency matrix S and the diagonal matrix as

$$S_{conv} = S - diag(l_1, l_2, , .., l_N).$$

4.2.2 Weisfeiler-Lehman algorithm

The Weisfeiler-Lehman (WL) algorithm, which was first proposed in 1968 to solve the *graph isomorphism problem* [84], has recently been widely applied in neural network models. For every vertex v_i, features from neighboring vertices are aggregated and computed to update its own feature, which is computed as follows:

$$x'(v_i) = \mathcal{AGG}(x(v_i), emb\{x(v_j)|v_j \in N(v_i)\}), \tag{1}$$

where $x(\cdot)$ and $x'(\cdot)$ are the original and updated features of vertices, respectively, and $N(v_i)$ denotes a set of neighboring vertices for vertex v_i, while *emb* is an embedding function based on neighborhood aggregation that concatenates features from neighboring vertices of v_i, and $\mathcal{AGG}$ is a custom function computing feature from the target vertex and its neighboring vertices. By implementing different functions, features can be updated in different ways. In addition, vertices can always have special features by sev-

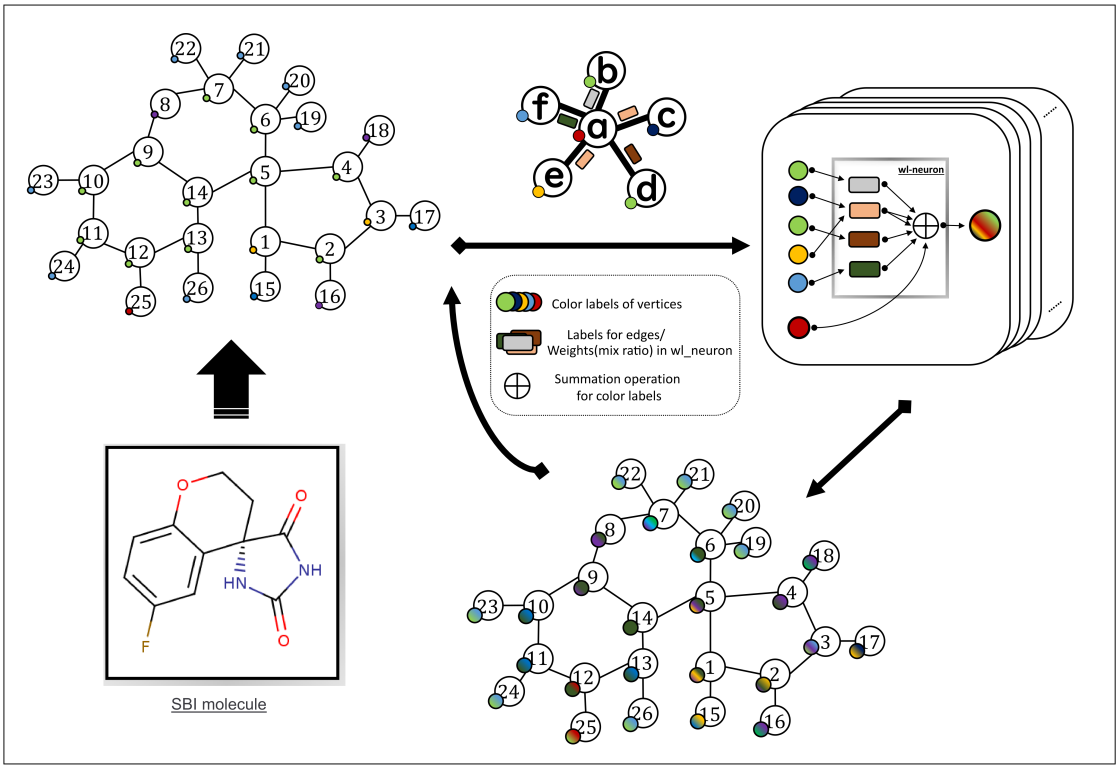


Figure 27: Weisfeiler-Lehman algorithm

eral repetitions even with large graphs. Subsequently, the isomorphism of two graphs can be analyzed by examining the different features of updated set of vertices.

As mentioned in **Section 4.2.1**, every vertex and edge have a *solid color* and label, respectively, and the *colors* of vertices are updated individually with the *colors* of the neighboring vertices and labels of the connected edge as illustrated in **Fig.27**. The aggregation function is called *mix*, which can *blend* multiple colors together. Here, the *mix ratio* is dependent on the labels of the edges between the updated and neighboring vertices. After several repetitions of the algorithm, every vertex has a unique *blended color* as the feature in the inhibitor molecule graph.

4.3 Model

The architecture of the proposed PISPKI model is shown in **Fig.28**. The model framework consists of four main parts: data preprocessing, WL Boxes, convolutional layers, and dense layers. First, each inhibitor molecule with N atoms is transformed to N pairs of feature matrices and structure adjacency matrices, where the ith atom is marked in the ith feature matrix to predict whether the corresponding atom is an interaction site. Note that the output of the model is assigned a value of 1 if the marked atom is predicted to be an interaction site. The feature matrices and structure adjacency matrices contain, respectively, atom and bond information of the molecules. Due to uncertainties about the number of atoms of the molecules, the sizes of the two matrices can be alterabled for different input data. Notably, zero-padding is not applied to satisfy all input data in the same size. After preprocessing, each pair of matrices is added to two submodules: WL Boxes and convolutional layers. The WL Boxes mainly process feature matrices using structure adjacency matrices as auxiliary information. Matrices with more significant features can be obtained from the output of WL Boxes; then the pooling layer processes new feature matrices into fixed-length vectors by applying the spatial pyramid pooling (SPP). By contrast, the convolutional layer processes structure adjacency matrices in which some atom information about the feature matrices is embedded in the diagonal elements, and the output is also processed by the SPP into a fixed-length vector. The resulting two vectors are concatenated as input to the dense layers for binary classification. Remarkably, the output vector from the pooling layer is obtained by combining the pooled results of the updated feature matrices and structure adjacency matrices, and the lengths of the vectors from the updated feature matrices are larger than those from the updated structure adjacency matrices, indi-

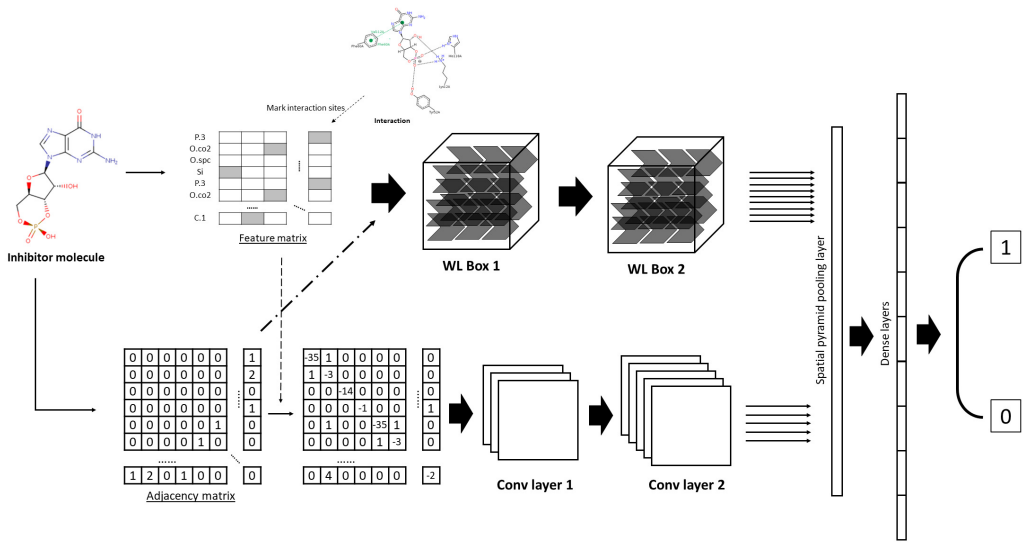


Figure 28: Architecture of the PISPKI model

cating that the output from the WL Boxes offers more information for the prediction of interaction sites by the classifier in the dense layer, thereby the WL Box module is the core of the PISPKI model.

4.3.1 WL Box

The WL Box, which is the core of the PISPKI model, is based on the WL algorithm as described in **Section 4.2.2**. A WL Box consists of $L \times T$ wl-neurons that are arranged by L layers and T time steps. Every layer contains T end-to-end wl-neurons and the feature matrix F is given an input for every first wl-neuron of the layers. Then, the wl-neuron updates the hidden state of the feature matrix in accordance with the WL algorithm and the structure adjacency matrix S is applied as supplementary information. The updated hidden state of the wl-neuron transfers to the next wl-neuron in each layer for $T - 1$ time steps. The output of the last wl-neuron from the layer is an output of the WL Box. Hence, L new feature matrices can be obtained that contain more significant feature information in a WL Box. Furthermore,

hidden states of feature matrices are updated in each layer individually, and there is no message exchange between layers in the box, as shown in **Fig.29**.

The hidden state of a row is recurrently updated by

$$h_l^{(t)}(i) = h_l^{(t-1)}(i) + \sum_{j=1}^{N} w_{l,S_{ij}}^{(t)} h_l^{(t-1)}(j), \tag{2}$$

where $h_l^{(t)}$ represents the current hidden state of the wl-neuron of the layer l at time step t, in which $l < L$ and $t < T$; $h_l^{(0)}$ denotes the initial state of the first neuron of the layer l, and $h_l^{(0)} = F$. Let $h_l^{(t)} = (h_l^{(t)}(1), h_l^{(t)}(2), ..., h_l^{(t)}(N))$, and the ith row vector of $h_l^{(t)}$ is represented by $h_l^{(t)}(i)$; $w_{l,S_{ij}}^{(t)}$, which is a trainable *switch weight* (real number) depending on the layer, time step, and bond type $S_{ij} \in \{0, 1, ..., N_e\}$, where $w_{l,0}^{(t)}$ equals *zero* in any wl-neuron regardless of the layers and time steps.

Finally, the hidden states of the last neuron of each layer l at time step T are combined to a tensor $\mathcal{F}[i, j, l]$ as the output of a WL Box after processing with the activation function by

$$\mathcal{F}[i, j, l] = \sigma(h_l^{(T)}[i, j]), \tag{3}$$

where $\mathcal{F}$ represents an output tensor of the WL Box, and $\mathcal{F}[-, -, l]$ denotes the lth *block* that is defined as a two-dimensional matrix consisting of all elements and the array from tensor $\mathcal{F}$ when the index is equal to l in *rank 3* in the tensor; σ is an activation function, and $h_l^{(T)}$ is the last hidden state of the lth layer at time step T. Besides, the structure adjacency matrix S is invariant during the process in the WL Box and can be completely delivered to the next module if needed.

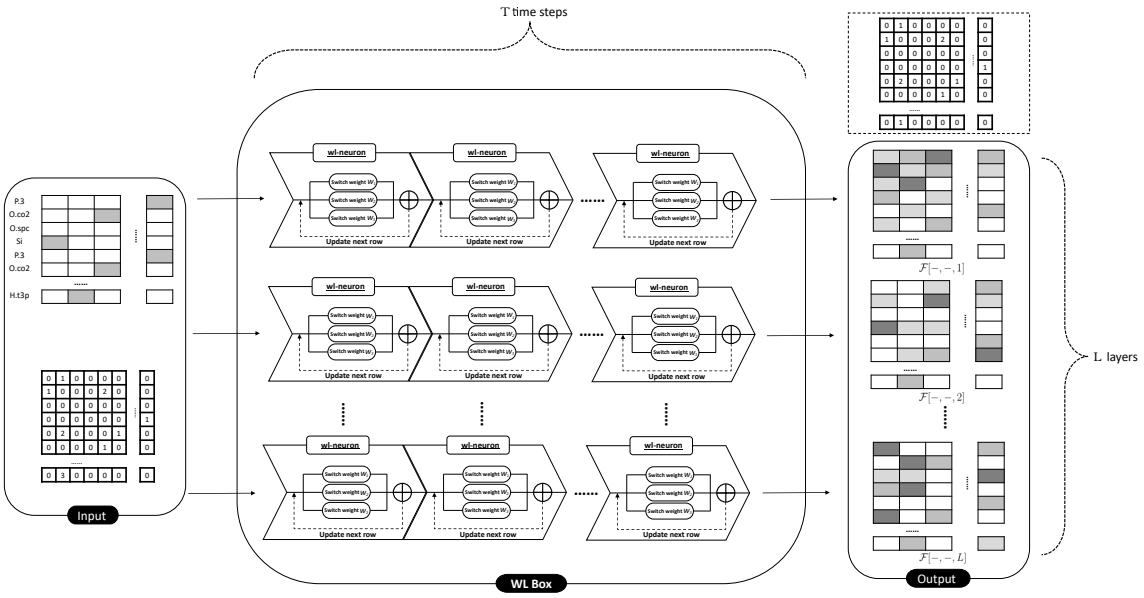


Figure 29: Structure of the WL Box

4.3.2 Multiple WL Box

Multiple WL Boxes can be implemented sequentially to further improve the feature matrix of the model. Two WL Boxes are assembled for the model as shown in **Fig.28**. Notably, the second WL Box, which differs from the first WL Box, receives a tensor $\mathcal{F}$ as the input rather than matrix F, and so on. To get around this issue, superscripts are sequentially assigned to the feature tensor $\mathcal{F}$, such as $\mathcal{F}^{(1)}, ..., \mathcal{F}^{(M)}$ and the layer number L, such as $L^{(1)}, ..., L^{(M)}$, denoting the output tensor and maximum layer number of the first to the Mth WL Box, respectively. The feature tensor of each WL Box is updated by

$$\mathcal{F}^{(m)} = \begin{cases} Concat_{l=1}^{L^{(m-1)}}(\Theta(\mathcal{F}^{(m-1)}[-,-,l])) & ,m \geq 2 \\ \Theta(F) & ,m = 1 \end{cases} \tag{4}$$

where $\mathcal{F}^{(m)}$ and $\mathcal{F}^{(m-1)}$ represent the output tensor of the mth and $m-1$th WL Box of the model, respectively; Θ denotes the WL update process

function in a WL Box, as defined by **Equations 2** and **3**; $\mathcal{F}^{(m-1)}[-,-,l] \in \mathbb{R}^{N_c \times N}$ is the lth *block* of $\mathcal{F}^{(m-1)}$, and F is the model input feature matrix. Every *block* $\mathcal{F}^{(m-1)}[-,-,l]$ from tensor $\mathcal{F}^{(m-1)}$ is assigned to the mth WL Box as an individual input, and all $L^{(m-1)}$ output tensors are concatenated into one tensor $\mathcal{F}^{(m)}$ for the following computation operation.

4.3.3 Pyramid spatial pooling

The spatial pyramid pooling (SPP) layer is applied to normalize the output from the WL Boxes and convolutional layers in this study, which is a novel and effective machine learning module proposed by He in 2015 [18]. Different from classical pooling modules, SPP is a type of extensive research for region of interest operation, which further works with different sizes of pooling kernels in a matrix, and then concatenates the pooling results to a vector as the output. This also applies to hand-crafted pooling regions [85] over scales of kernels that are dependent on different sizes of input matrices and adopts the spatial pyramid operation to obtain more comprehensive pooled feature maps, which are then converted to a fixed length vector (**Fig.30**).

A spatial pyramid consists of multiple stages, and each stage runs a pooling operation using the corresponding pooling coefficient. The notation k represents the pooling coefficient with $k = 1, ..., K$, where K is the stage number of a spatial pyramid. During each operation by the spatial pyramid, a hand-crafted kernel is applied, which yields precise $k \times k$ output from the inputted two-dimensional matrix. Due to differences in kernel size, each input is extended to

$$p_0^k = \Xi_k(F), \tag{5}$$

where F is a *block* of a tensor or matrix of $N_c \times N$ and Ξ_k is the matrix

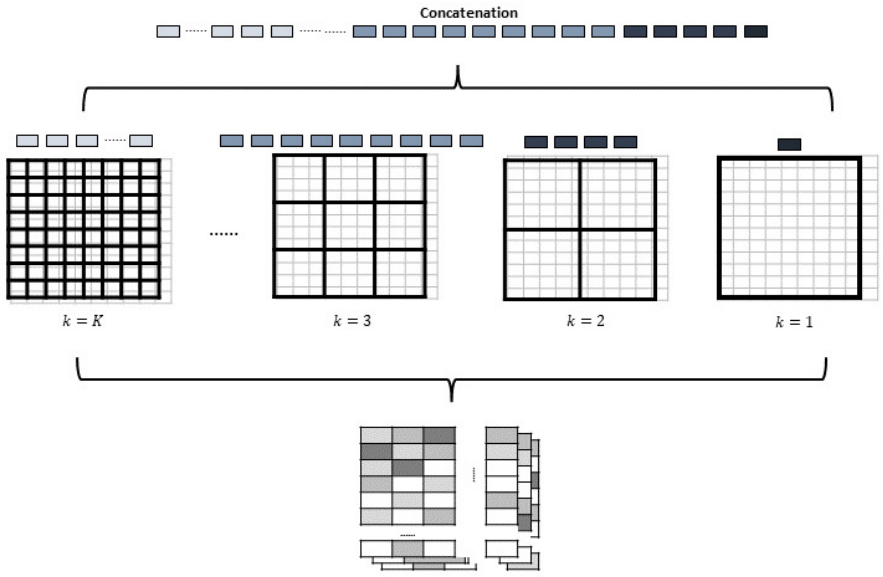


Figure 30: The structure of spatial pyramid pooling

extension function for pooling coefficient k. By applying the function, F is extended to a matrix $p_0^k = \mathbb{R}^{k\cdot\lceil\frac{N_c}{k}\rceil\times k\cdot\lceil\frac{N}{k}\rceil}$, and all extended elements are equal to 0.

Here, the SPP layer receives two tensors $\mathcal{F}$ and $\mathcal{S}$ from the last WL Box and the convolutional layer. Two spatial pyramids are constructed (with stage numbers K_F and K_S) to individually compute the two tensors. The pooling operation works with every *block* K_F and K_S times for the input tensors $\mathcal{F}$ and $\mathcal{S}$, respectively. Maximum SPP is applied to the output tensor $\mathcal{F}$ of the WL Box. For the kth stage of the spatial pyramid, each element of the xth column and yth row of matrix p_f^k is computed by

$$p_f^k[x,y] = \sum_{i=1}^{I}\sum_{j=1}^{J} p_0^k[I\cdot(x-1)+i, J\cdot(y-1)+j]/IJ, \tag{6}$$

where k denotes the pooling coefficient, $k \in \{1, ..., K_F\}$; p_0^k is an extended matrix computed by **Equation 5**; $p_0^k[X, Y]$ denotes the element of the Xth column and Yth row; (I, J) is the hand-crafted region in the kth stage, where $I = \lceil \frac{N_c}{k} \rceil$ and $J = \lceil \frac{N}{k} \rceil$.

Similarly, the average SPP was applied to the output tensor $\mathcal{S}$ from the convolutional layer. For the kth stage of the spatial pyrmaid, each element of the xth column and yth row of matrix p_s^k is computed by

$$p_s^k[x, y] = \max(p_0^k[I \cdot (x-1) : I, J \cdot (y-1) : J]), \tag{7}$$

where k denotes the pooling coefficient, $k \in \{1, ..., K_S\}$; p_0^k is an extended matrix computed by **Equation 5**, $p_0^k[X : I, Y : J]$ denotes a submatrix collecting elements from Xth to $(X + I)$th columns and Yth to $(Y + J)$th rows; I and J constitute the hand-crafted region of the kth stage, where $I = \lceil \frac{N_c}{k} \rceil$ and $J = \lceil \frac{N}{k} \rceil$.

Finally, all elements are collated from the pooling matrices to an output vector as

$$P = Concat(\Phi(p_f^1), \Phi(p_f^2), ..., \Phi(p_f^{K_F}), \Phi(p_s^1), ..., \Phi(p_s^{K_S})), \tag{8}$$

where Φ is a function that converts a matrix to a vector, such as $\Phi(p_f^2) = (p_f^2[1, 1], p_f^2[1, 2], p_f^2[2, 1], p_f^2[2, 2])$.

Then, the output of the SPP layer is sent to a binary classifier for interaction site prediction by the dense layer.

4.4 Experiments

4.4.1 Dataset preprocess

Protein-ligand complexes and interaction sites were collected from the *sc-PDB* three-dimensional database of ligandable binding sites [86] and grouped by protein UniProt identifications from the *Protein Data Bank* [87]. In total, 1,064 protein-ligand complexes datasets of 22 protein kinases were extracted and categorized into 11 corresponding kinase classes as shown in **AppendixTable S1**. A program was developed to convert the *mol2 file* to a model input file consisting of the feature matrix $F \in \{0,1\}^{35\times N}$ and the structure adjacency matrix $S \in \{0,1,2,3,4\}^{N\times N}$ for each inhibitor molecule consisting of N atoms from the protein-ligand complex. According to the *mol2* format, there are 35 atom types and eight bond types. The categorical features of every atom were *one-hot* encoded as a *color* label and aligned with the feature matrix F. The bond defined as *single*, *triple*, *dummy*, *unknown*, and *not connected* were classified as TYPE 1, a *double* bond as TYPE 2, an *amide* bond as TYPE 3, and an *aromatic* bond as TYPE 4. The TYPE 1 category consists of five bond types (i.e., *single*, *triple*, *dummy*, *unknown*, and *not connected*) in the dataset. Here, the *single* bond is the most common bond type, and the remaining four bond types are rare. The structure adjacency matrix S is a record of the connection relationships between two atoms and their corresponding BOND TYPE of the inhibitor molecule. TYPE 0 can be used for any two atoms with bond types that are not mentioned in the the *mol2 file*.

4.4.2 Experiment setup

An individual prediction model was established for each class of kinases. An inhibitor molecule with N atoms provides N data pairs (F, S) by assigning a specific mark to the label of each atom of feature matrix F. If a marked atom binds with a residue of the kinase, the corresponding output assigned a value of 1, otherwise, 0. The binding types between atoms and residues were ignored, as the binding state was the focus of this study. Each of the original datasets was expanded to the one with at least 2,560 positive and 2,560 negative samples by using the method described in **Section 4.4.3**, and the resulting expanded rates are shown in the last column of **Table 2**. The dataset was randomly split into three parts: one tenth positive/one tenth negative datasets into the test dataset, one tenth positive/one tenth negative datasets into the validation dataset, and rest of datasets into the training dataset, where three datasets were totally non-overlapping. Training datasets were used to train PISPKI models of each kinase class for several epochs, and models were evaluated by validation datasets at each epoch after training. Furthermore, *bootstrapping* was applied to the training and validation datasets to uniformly assign samples at each epoch. The program was developed with *PyTorch* [88]. As shown by the model setup in **Table 1**, *early-stopping* [89] was set to 5 epochs to avoid overfitting issues and the accuracy of the sixth to last validation was recorded. After training was completed, the model was evaluated with the testing dataset.

Noise Elimination Multiple protein-ligand complexes consisted of the same protein (kinase) and ligand (inhibitor), but with different interaction sites. However, unique confusing events can occur, such as the existence of a kinase inhibitor with two *crystal structures* (α and β) and an atom of the

Components	**Parameters**
WL Box 1	3 layers × 3 time steps
WL Box 2	3 layers × 3 time steps
Conv-layer 1	1 input channel 2 output channel 3 × 3 kernels
Conv-layer 2	2 input channel 5 output channel 3 × 3 kernels
SPP cofficient	feature matrix: 10 Structure matrix: 3
Dense layers	2000 neurons × 5 layers
Activation function	Leaky ReLU
Ealry stopping	5 epochs
Dropout	0.05
Turn size	2048×2(Training dataset) 256×2(Validation dataset)
Batch size	16

Table 1: Model setup

inhibitor that binds with a residue of the kinase of *crystal structure* α but does not bond with any residue of *crystal structure* β.

To eliminate this type of noisy data, the notation $\mathcal{B}_I^{(K)}$ was used to denote inhibitor molecule I having M crystal structures on kinase K to represent a set consisting of all atoms binding with K. With $B_m^{(K,I)}$ designating a binding atom set for one of the crystal structures composed of inhibitor I and kinase K, the following definition is obtained:

$$\mathcal{B}_I^{(K)} = \bigcup_{m=1}^{M} B_m^{(K,I)}.$$

Then, for atom i of the inhibitor molecule I, the interaction state $Y_i^{(K,I)}$

for kinase K is determined by

$$Y_i^{(K,I)} = \begin{cases} 1, & i \in \mathcal{B}_I^{(K)}, \\ 0, & otherwise. \end{cases}$$

4.4.3 Dataset expander program

Due to the limited number of original datasets (**Appendix Table S2**), a dataset expander algorithm was developed inspired by the expansion method widely applied with image recognition datasets. A *seed* is randomly assigned to the *reindex rows* or *columns* of matrices, and the *reindex* operation Ψ does not change the structure of the inhibitor molecule but creates a different input data pair. An example of the structure adjacency matrix expansion is shown in **Appendix Fig.S2**. Each input pair (F, S) from the original dataset is modified to a new pair utilizing the same *seeds* for F and S while maintaining the same output Y as follows:

$$\begin{cases} F' &= \Psi_{r,c}(F, seed) \\ S' &= \Psi_r(S, seed) \\ Y' &= Y \end{cases}$$

where F', S' and Y' are the created feature matrix, structure adjacency matrix, and output, respectively. In addition, $\Psi_{r,c}$ indicates that the *reindex* operation was applied to both *rows* and *columns*, whereas Ψ_r indicates application to rows only. By utilizing different *seeds* with an original pair, multiple different sample pairs can be obtained up to $P_N^N = N!$, where N is the atom number of the inhibitor molecule. The batch process method of the expander

Algorithm 1 Dataset expander program

Input: Orignal feature dataset F; original structure dataset S; expansion rate r
Output: Enlarged feature dataset F_e; enlarged structure dataset S_e
original dataset length $l \leftarrow len(F)$
random seed $seed \leftarrow random()$
for $x = 1$ to $l \times r$ **do**
new feature matrix $f \leftarrow reindex_{row}(F[x \mod l], seed)$
$f \leftarrow reindex_{column}(f, seed)$
$F_e.append(f)$
$S_e.append(reindex_{row}(S[x \mod l], seed))$
if $x \mod 10 == 0$ **then**
$seed \leftarrow random()$
end if
end for
return F_e, S_e

program is shown in **Algorithm 1**. As mentioned in **Section 4.4.1**, original datasets transformed from the *mol2* format were collected with arrays of atoms in a particular order. The dataset expander program can also potentially support the model to improve compatibility with datasets collected from formats other than *mol2*.

4.5 Results

4.5.1 Baseline experiment

The performance of the proposed PISPKI model was comprehensively evaluated by comparison with Support Vector Machine (SVM) and Convolutional Neural Network (Conv-Net) models as baselines, where the SVM model applies the *radial basis function* kernel and Conv-Net has a traditional architecture consisting of two convolutional layers and a fully connected layer. Feature matrices with *zero-padding* were used as input for the baseline models.

Kinase	Number of PLC*	Subclass	N_{max}	Conv-Net (%)	SVM (%)	PISPKI(%)		Expansion rate(p/n)
						Validation	Test	
3-phosphoinositide-dependent protein kinase	41	1	73	79.0	74.0	84.0	84.7	8/3
Aurora kinase	58	1	81	56.7	70.0	79.1	80.8	10/3
Circadian clock protein kinase	16	1	44	50.0	85.0	93.0	91.5	190/89
Cyclin-dependent kinase	280	1	67	76.0	68.3	86.7	85.1	2/1
Death-associated protein kinase	28	1	62	68.3	74.0	80.9	83.5	27/11
Dual specificity mitogen-activated protein kinase kinase	24	1	58	71.7	63.0	84.0	81.9	8/5
Glucokinase	20	1	55	78.3	75.0	85.5	85.5	25/9
Glycogen synthase kinase	40	1	74	61.7	67.0	84.4	85.5	11/3
Serine/threonine-protein kinase	197	4	69	63.3	67.3	85.4	85.0	2/1
Tyrosine-protein kinase	99	5	93	56.5	65.9	84.4	86.7	3/1
Proto-oncogene tyrosine-protein kinase	17	1	76	73.3	65.0	83.6	79.5	180/60
Mitogen-activated protein kinase	244	4	87	78.8	64.9	86.9	87.5	1/1

*PLC=protein-ligand complex

Table 2: Comparison of the validation and test (%) performance of different models

The highest accuracy of 10 repeated experiments was recorded. Comparison of the proposed PISPKI model and the two baseline models is shown in **Table 2**.

The accuracy of the PISPKI model to predict whether an atom from an inhibitor molecule is an interaction site or not mostly ranged from 83% to 86% for the different kinase classes, which was notably better than that of the two baseline models. In addition, both the Conv-Net and SVM models were unstable with different datasets of kinase classes, whereas the proposed model was not. Although the accuracy for the *Circadian clock protein kinase* was high, the prediction accuracy of the model is not necessarily high because the corresponding dataset contained only 16 protein-ligand complexes.

However, the expansion rate has no effect on the prediction of the interaction site, with the exception of extreme situations, such as the *Circadian clock protein kinase* mentioned above. Nonetheless, the performance of the model can be improved by applying a small number of expansion operations, as the accuracies of datasets with expansion rates of less than 3, such as the *mitogen-activated protein kinase*, *tyrosine-protein kinase*, *serine/threonine-protein kinase*, and *cyclin-dependent kinase*, are stable at about 86%.

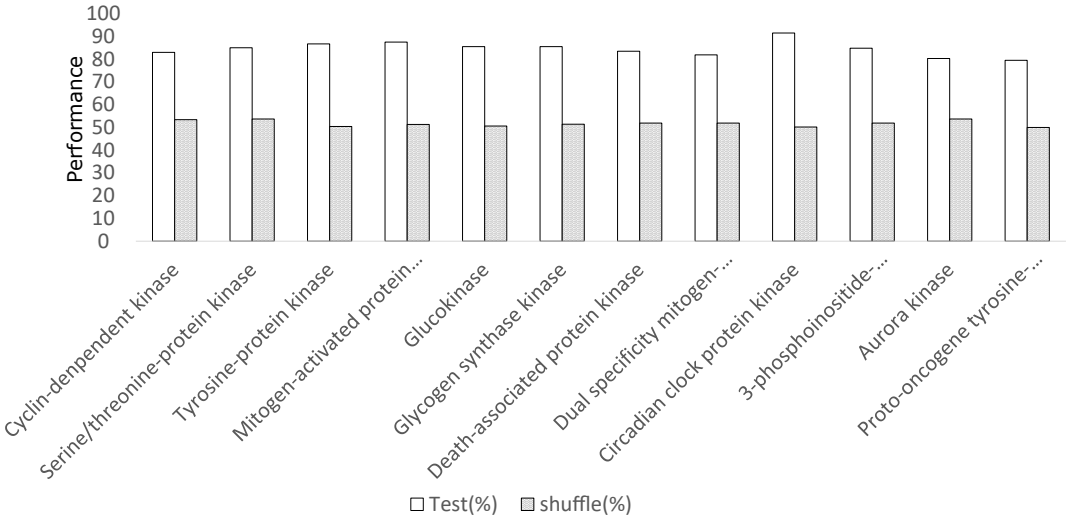


Figure 31: Comparison of the performance of the shuffled and testing datasets

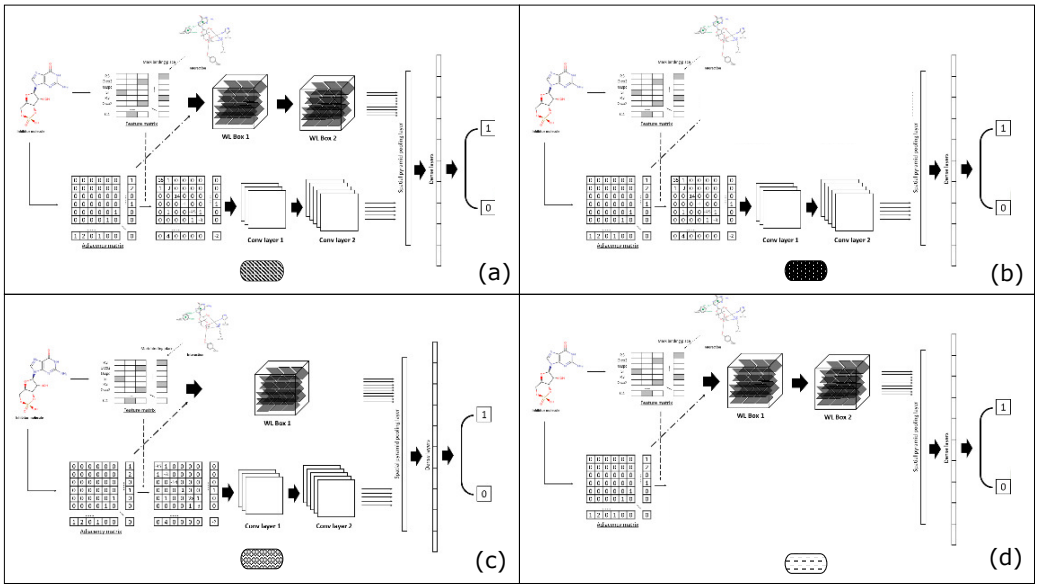
4.5.2 Shuffle dataset testing

The effectiveness of the PISPKI model was further assessed with shuffled datasets. Due to the limited number, portions of the validation datasets were randomly extracted and the interaction sites were shuffled to create shuffled datasets. Consider two *cases*: *(1)* the PISPKI model could still predict the interaction sites of shuffled datasets with an accuracy equal to or greater than that of the testing datasets; and *(2)* the model is not compatible with shuffled datasets or the accuracy is obviously degraded. The baseline accuracy was set to 50% to denote the state "cannot work", as such a situation is a binary classification issue. *Case (1)* suggests the model is compatible with both correct and incorrect data, indicating a problematic state, whereas *case (2)* confirmed the effectiveness of the model. The performances of the shuffled and testing datasets for each kinase class are compared in **Fig.31**. The PISPKI model is incompatible with shuffled datasets, thereby validating its effectiveness.

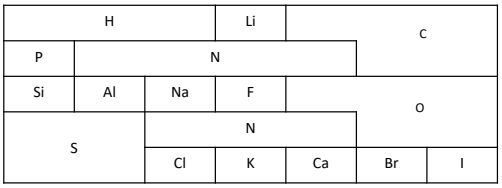
4.5.3 Ablation study

To ensure and discuss the necessity of each part of the PISPKI model, an ablation experiment was designed in which the performance of the model was assessed by removing components. In the experiment, only four typical kinase class datasets with expansion rates of less than 3 were collected. Then, (1) two WL Boxes; (2) one WL Box; (3) and the Conv-layers were abandoned, and (4) 35 atom subtypes were merged into 16 *colors* by combining the same types of chemical elements to successively construct four incomplete models, which are illustrated in **Fig.32**. The performance of the incomplete models was compared to that of the PISPKI model(**Fig.33**).

The performance of the PISPKI model was significantly compromised by removing two WL Boxes from most datasets, which obviously decreased the accuracy. In addition, the model was incompatible with the *mitogen-activated protein kinase* dataset, thereby confirming that the WL Box is the core of the PISPKI model. As shown in the third column of each dataset in the figure, reducing the number of WL Boxes to one had very limited influence on the model. However, compared with the full model, the performance of the truncated model was improved by adding extra WL Boxes. The convolutional layers seem to be an insignificant component of in most datasets, which still suggests potential advantages. Notably, the convolutional layers only process original structures and feature label information as mentioned in **Section 4.3**. Although the WL Boxes process the feature and structure information more exquisitely, the original information processed by the convolutional layers facilitates inference of the interaction sites more accurately in complicated cases. In the last ablation experiment, the necessity of feature richness, which represents the quality of each feature, was tested. For this evaluation, datasets from *mol2* files were collected, which encoded atoms in



H	H.spc	H.t3p	Li	C.1	C.2	C.3
P.3	N.4	N.3	N.2	N.1	C.cat	C.ar
Si	Al	Na	F	O.t3p	O.spc	O.co2
S.2	S.3	N.am	N.ar	N.p13	O.2	O.3
S.o	S.o2	Cl	K	Ca	Br	I



(e) 35 → 16 colors

Full model Without two WL Boxes Only use one WL Box Without conv-layers 16 color

Figure 32: Ablation experiment setting

the *SYBYL* format that were further divided into 35 subtypes (*color*) and combined into 16 types (*color*) based on chemical elements as illustrated in **32(e)**. By the combination operation, the performance of the model with the different datasets decreased by various degrees. The results not only highlight the importance of *SYBYL* atom types but also serve as a reminder that the performance of the PISPKI model can be improved by enhancing feature richness.

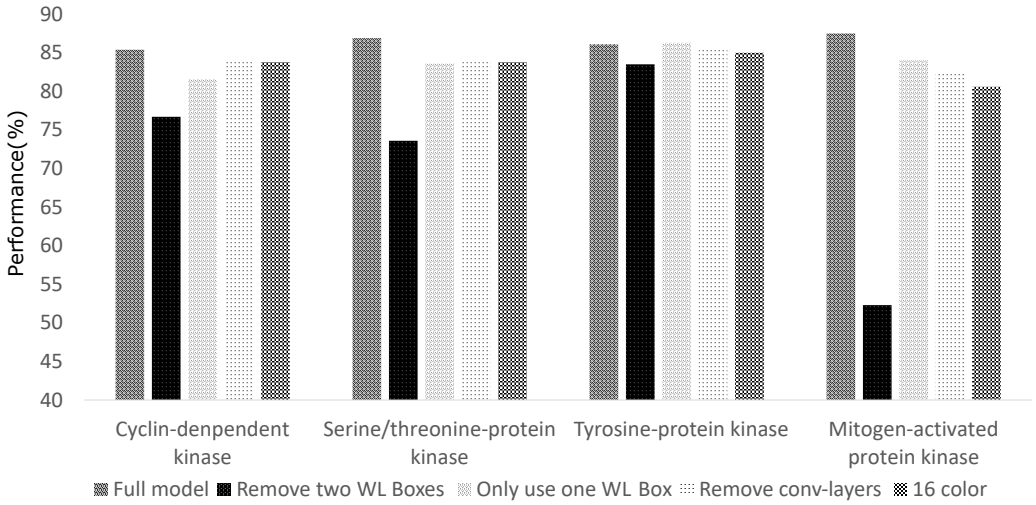


Figure 33: Ablation analysis

In the ablation experiments, incomplete models were applied to examine differences in performance loss observed from the datasets. The effects on the *cyclin-dependent kinase* and *serine/threonine-protein kinase* datasets were very limited by applying incomplete models. However, the *tyrosine-protein kinase* and *mitogen-activated protein kinase* datasets had extremely low and high impacts, respectively. As mentioned above, the PISPKI model aims to solve the issue with interaction site prediction. However, there were multiple different sub-issues due to the kinase class, which is another reason why the accuracy of the baseline models was extremely variable with different kinase class datasets (**Table 2**).

4.6 Discussions

In this study, a novel machine learning model (i.e., WL algorithm-based GCN network) was designed and developed to predict interaction sites of protein inhibitors in phosphorylation. The accuracy of the model was consistently

83% to 86%, which can be greatly improved by applying datasets with low expansion rates compared to the two baseline models. The model performance can be improved by the addition of feature richness. At present, features are transformed from inhibitor molecules based on *SYBYL* atom types, which contain more information than chemical elements. More information about atoms can be collected to enhance the richness of features such as *radius*, *atomic mass*, *formal charge*, and *aromaticity*. In addition, the protein kinase residue information should also be used as input to the model. Furthermore, the limitation of datasets effects the model performance. This research was not only limited by input feature richness but also by the small number of datasets because there have been relatively few investigations to identify the interaction sites of inhibitor molecules. The spatial pyramid pooling (SPP) module facilitated compatibility of the model with inhibitor molecules having different number of atoms. Furthermore, the importance of the WL Box was confirmed by the ablation study (**Section 4.5.3**), which showed that the addition of multiple WL Boxes can enhance performance. Although applying a complicated model on a simple issue is not recommended owing to potential performance degradation because of excess trainable parameters, the PISPKI model can predict most interaction sites and solve other complicated biology issues. Hence, stable model performance is absolutely critical.

Chapter 5

Conclusion

5 Conclusion

In this thesis, we studied computational methods for analyzing graph-structured biological data with focusing on the comparison of pseudoknotted RNA secondary structures and prediction of interaction sites of protein kinase inhibitors.

The proposed approach for comparison pseudo-knotted RNA secondary structures is comprised of a novel tree representation by topological centroid identification and their comparison methods based on the tree edit distance, where a given graph representing an RNA secondary structure is transformed to a tree rooted at one of the vertices constituting the topological centroid that is identified by removing cycles with **PEELING** processing for the graph. When comparing tree-represented RNA secondary structures collected from a public database using the tree edit distance and functional gene groups defined by Gene Ontology (GO), the proposed method showed better clustering results according to their GOs than canonical RNA sequence-based comparison. In addition, we also reported a case that the combination of the tree edit distance and the sequence edit distance showed a better classification of the pseudoknotted RNA secondary structures. In the study, only partial structures (i.e., pseudo-knots) for RNA molecules were analyzed. The approach should be extended for comparison of whole RNA molecules to further analyze secondary structures. Moreover, the costs of substitution operations of the tree edit distance between different labels (nucleobases) are uniform, but the four types of nucleobases have differences, and those nucleobases of RNAs should not be simply regarded as just labels [‖]. In this study, a novel approach was initially applied to the RNA secondary structure issue, so nu-

[‖] Objects of most common computational models have simple features that usually do not have deep relationships with each other. But, nucleobases are not. For example, base A has closer relationships with base U compared to base C or base G based on the structure of the molecule.

cleobases were processed as labels. Future work should attach importance to differences between four nucleobases, and applying various costs of substitution operations between nucleobases should be considered. Furthermore, machine learning methods (e.g. graph convolutional neural network) should also be applied to solve the RNA secondary structure issue.

To solve the binding mechanism issue, a novel machine learning module (i.e., the WL Box) was designed and assembled to the Prediction of Interaction Sites of Protein Kinase Inhibitors (PISPKI) model, which is a graph convolutional neural network (GCN) to predict the interaction sites of protein kinase inhibitors. The WL Box is a novel module based on the well-known Weisfeiler-Lehman algorithm, which assembles multiple switch weights to effectively compute graph features. The PISPKI model was evaluated by testing with shuffled datasets and ablation analysis using 11 kinase classes. The accuracy of the PISPKI model varied from 83% to 86%, demonstrating the superior performance compared to two baseline models. The effectiveness of the model was confirmed by testing with shuffled datasets. Furthermore, the performance of each component of the model was analyzed via the ablation study, which demonstrated that the WL Box module was critical. There are some rooms for improvements of the proposed method. For examples, it is useful to utilize more information about atoms to enhance the richness of features such as *radius*, *atomic mass*, *formal charge*, and *aromaticity*. In addition, the protein kinase residue information should also be used as input to the model.

Appendix

Algorithm 2 DCEL converter

Input: list of nucleobase labels L; list of pairs P

Output: vertex List V; half-edge list E; face list F

1: **for** l in L:

2: v=new vertex(); v.label=l; V.add(v)

3: **end for**

4: he=new half_edge();he.targetV=V[0];E.add(he)

5: **for** i in 1...V.len-1:

6: he=new half_edge();he.targetV=V[i];he.prevE=V[i-1];he.prevE.nextE=V[i];E.add(he)

7: **end for**

8: he=new half_edge();he.targetV=V[-2];he.prevE=E[-1];he.twinE=E[-1];he.E.add(he)

9: **for** i in V.len-2...0:

10: he=new half_edge();he.targetV=V[i];he.prevE=V[i-1];he.prevE.nextE=V[i];

11: he.twinE=he.prevE.twinE.nextE;E.add(he);

12: **end for**

13: **while** *true*:

14: **if** (P[e.targetV]!=0)

15: e2=*search_pair_half_edge* 16: he1=new half_edge();he2=new half_edge()

he1.twinE=he2;he2.twinE=he1;

17: *switch* (Direction):

18: *case*:1

19: he1.targetV=e.targetV;he2.targetV=e2.targetV

20: he1.nextE=e2.twinE;e2.twinE=he1;

21: he1.prevE.prevE=e.nextE.twinE;e.nextE.twinE.nextE=he1;

22: he2.nextE=e.twinE;e.twinE.prevE=he2;

23: he2.prevE=e2.nextE.twinE;e2.nextE.twinE.nextE=he2;

24: E.add(he1);E.add(he2)

25: e=e.prevE 26: *case*:2

27: he1.targetV=e2.targetV;he2.targetV=e.targetV

28: he1.nextE=e.twinE;e.twinE.prevE=he1;

29: he1.prevE=e2.nextE.twinE;e2.nextE.twinE.nextE=he1;

30: he2.nextE=e2.twinE;e2.twinE.prevE=he2;

31: he2.prevE=e.nextE.twinE;e.nextE.twinE.nextE=he2;

32: E.add(he1);E.add(he2)

33: e=e.nextE

34: Direction_swap_judge()

35: *end switch*

36: **end if**

37: **if** e.targetV=V[-1]:

38: *break*

39: **end if**

40: **end while**

41: F=loop_identify()

42: repair_program()

43: **return** V,E,F

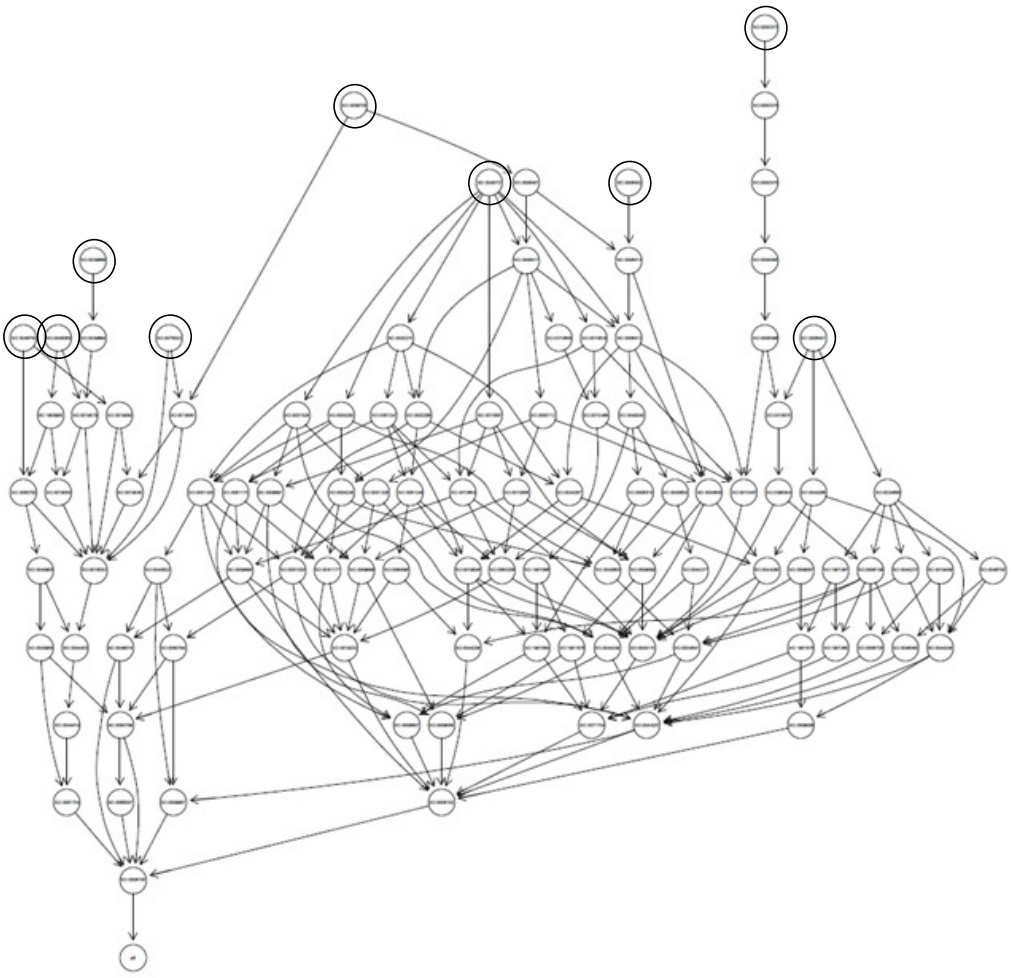


Figure S1: Acyclic relationship graph for nine gene ontology terms

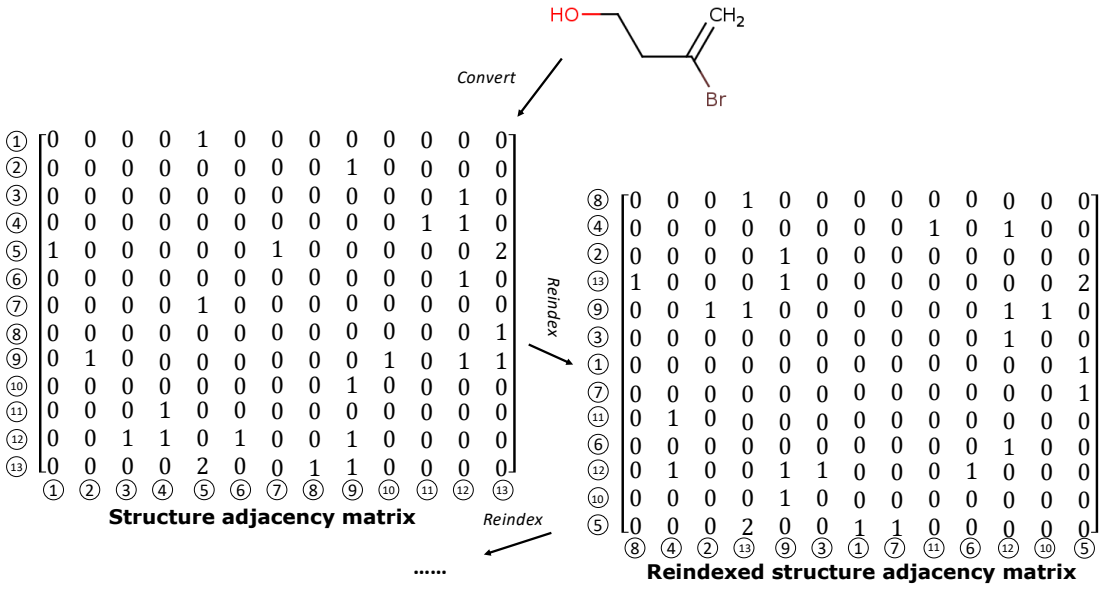


Figure S2: An example of the expander program

Kinase class	**UniPort AC**
3-phosphoinositide-denpendent protein kinase	O15530
Aurora kinase	O14965
Circadian clock protein kinase	Q79PF4
Cyclin-denpendent kinase	P24931
Death-associated protein kinase	P53355
Dual specificity mitogen-activated protein kinase kinase	Q02750
Glucokinase	P35557
Glycogen synthase kinase	P49841
Serine/threonine-protein kinase	O96017 P51955 P11309 O14757
Tyrosine-protein kinase	O60674 P00519 P43405 P06239 P08631
Proto-oncogene tyrosine-protein kinase	P00523
Mitogen-activated protein kinase	P28482 Q16539 P47811 P53779

Table S1: Kinase classes

Kinase class	**Dataset quality** (p/n)
3-phosphoinositide-denpendent protein kinase	362/1060
Aurora kinase	273/1146
Circadian clock protein kinase	14/29
Cyclin-denpendent kinase	2521/7377
Death-associated protein kinase	98/253
Dual specificity mitogen-activated protein kinase kinase	337/596
Glucokinase	104/294
Glycogen synthase kinase	329/1038
Serine/threonine-protein kinase	1796/5654
Tyrosine-protein kinase	1003/3419
Proto-oncogene tyrosine-protein kinase	146/460
Mitogen-activated protein kinase	2826/7324

Table S2: Dataset quality

Figure contents

Table contents

References

[1] Nguyen, T. T., Abdelrazek, M., Nguyen, D. T., Aryal, S., Nguyen, D. T., & Khatami, A. (2020). Origin of novel coronavirus (COVID-19): a computational biology study using artificial intelligence. *bioRxiv*:10.1101/2020.05.12.091397.

[2] Noor, M. A., Raza, A., Arif, M. S., Rafiq, M., Nisar, K. S., Khan, I., & Abdelwahab, S. F. (2022). Non-standard computational analysis of the stochastic Covid-19 pandemic model: an application of computational biology. *Alexandria Engineering Journal*, 61(1), 619-630.

[3] Sumon, T. A., Hussain, M., Hasan, M., Hasan, M., Jang, W. J., Bhuiya, E. H., ... & Lee, E. W. (2021). A revisit to the research updates of drugs, vaccines, and bioinformatics approaches in combating COVID-19 pandemic. *Frontiers in Molecular Biosciences*, 7, 493.

[4] Cannataro, M., & Harrison, A. (2021). Bioinformatics helping to mitigate the impact of COVID-19-Editorial. *Briefings in Bioinformatics*, 22(2), 613-615.

[5] Trunova, S. A., Dubatolova, T. D., & Omel'ianchuk, L. V. (2001). Phase-specific elements of the regulatory zone of the Drosophila melanogaster string gene. *Genetika*, 37(12), 1616-1620.

[6] Tsuruoka, Y., McNaught, J., Tsujii, J. I. C., & Ananiadou, S. (2007). Learning string similarity measures for gene/protein name dictionary look-up using logistic regression. *Bioinformatics*, 23(20), 2768-2774.

[7] Brijder, R., Langille, M., & Petre, I. (2007). A string-based model for simple gene assembly. *In International Symposium on Fundamentals of Computation Theory* (pp. 161-172). Springer, Berlin, Heidelberg.

[8] Levenshtein, V. I. (1966, February). Binary codes capable of correcting deletions, insertions, and reversals. *In Soviet Physics Doklady* 10(8), pp. 707-710.

[9] Zhang, K., & Shasha, D. (1989). Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6), 1245-1262.

[10] Demaine, E. D., Mozes, S., Rossman, B., & Weimann, O. (2009). An optimal decomposition algorithm for tree edit distance. *ACM Transactions on Algorithms*, 6(1), 1-19.

[11] Sidorov, G., Gòmez-Adorno, H., Markov, I., Pinto, D., & Loya, N. (2015). Computing text similarity using tree edit distance. *In 2015 Annual Conference of the North American Fuzzy Information Processing Society* held jointly with 2015 5th World Conference on Soft Computing (WConSC) (pp. 1-4). IEEE.

[12] Fukagawa, D., Tamura, T., Takasu, A., Tomita, E., & Akutsu, T. (2011). A clique-based method for the edit distance between unordered trees and its application to analysis of glycan structures. *BMC bioinformatics*, 12(1), 1-9.

[13] Bille, P. (2005). A survey on tree edit distance and related problems. *Theoretical Computer Science*, 337(1-3), 217-239.

[14] Riba, P., Fischer, A., Lladòs, J., & Fornès, A. (2021). Learning graph edit distance by graph neural networks. *Pattern Recognition*, 120, 108132.

[15] Cort‘es, X., Conte, D., Cardot, H., & Serratosa, F. (2018). A deep neural network architecture to estimate node assignment costs for the graph edit distance. *In Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)* (pp. 326-336). Springer, Cham.

[16] Neuhaus, M., & Bunke, H. (2004). A probabilistic approach to learning costs for graph edit distance. In Proceedings of *the 17th International Conference on Pattern Recognition*, 2004. ICPR 2004. (Vol. 3, pp. 389-393). IEEE.

[17] 徐周波, 张鹃, 宁黎华, & 古天龙. (2018). 图编辑距离概述. 计算机科学, 45(4), 11-18.

[18] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9), 1904-1916.

[19] Hofacker, I. L., Fontana, W., Stadler, P. F., Bonhoeffer, L. S., Tacker, M., & Schuster, P. (1994). Fast folding and comparison of RNA secondary structures. *Monatshefte f“ur Chemie/Chemical Monthly*, 125(2), 167-188.

[20] Dulucq, S., & Tichit, L. (2003). RNA secondary structure comparison: exact analysis of the Zhang-Shasha tree edit algorithm. *Theoretical Computer Science*, 306(1-3), 471-484.

[21] Giedroc, D. P., Theimer, C. A., & Nixon, P. L. (2000). Structure, stability and function of RNA pseudoknots involved in stimulating ribosomal frameshifting. *Journal of molecular biology*, 298(2), 167-185.

[22] Chen, J. H., Le, S. Y., & Maizel, J. V. (1992). A procedure for RNA pseudoknot prediction. *Bioinformatics*, 8(3), 243-248.

[23] Akutsu, T. (2000). Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots. *Discrete Applied Mathematics*, 104(1-3), 45-62.

[24] Jiang, T., Lin, G., Ma, B., & Zhang, K. (2002). A general edit distance between RNA structures. *Journal of Computational Biology*, 9(2), 371-388.

[25] Blin, G., Denise, A., Dulucq, S., Herrbach, C., & Touzet, H. (2008). Alignments of RNA structures. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 7(2), 309-322.

[26] Smit, S., Rother, K., Heringa, J., & Knight, R. (2008). From knotted to nested RNA structures: a variety of computational methods for pseudoknot removal. *RNA*, 14(3), 410-416.

[27] Andronescu, M., Condon, A., Hoos, H. H., Mathews, D. H., & Murphy, K. P. (2007). Efficient parameter estimation for RNA secondary structure prediction. *Bioinformatics*, 23(13), i19-i28.

[28] Smit, S., Yarus, M., & Knight, R. (2006). Natural selection is not required to explain universal compositional patterns in rRNA secondary structure categories. *RNA*, 12(1), 1-14.

[29] Poolsap, U., Kato, Y., & Akutsu, T. (2009). Prediction of RNA secondary structure with pseudoknots using integer programming. *BMC Bioinformatics*, 10(1), 1-11.

[30] Sato, K., Kato, Y., Hamada, M., Akutsu, T., & Asai, K. (2011). IPknot: fast and accurate prediction of RNA secondary structures with pseudoknots using integer programming. *Bioinformatics*, 27(13), i85-i93.

[31] Chiu, J. K. H., & Chen, Y. P. P. (2017). A comprehensive study of RNA secondary structure alignment algorithms. *Briefings in Bioinformatics*, 18(2), 291-305.

[32] Hochsmann, M., Voss, B., & Giegerich, R. (2004). Pure multiple RNA secondary structure alignments: a progressive profile approach. IEEE/ACM transactions on *Computational Biology and Bioinformatics*, 1(1), 53-62.

[33] Evans, P. A. (2006). Finding common RNA pseudoknot structures in polynomial time. *In Annual Symposium on Combinatorial Pattern Matching* (pp. 223-232). Springer, Berlin, Heidelberg.

[34] M'ohl, M., Will, S., & Backofen, R. (2008). Fixed parameter tractable alignment of RNA structures including arbitrary pseudoknots. *In Annual Symposium on Combinatorial Pattern Matching* (pp. 69-81). Springer, Berlin, Heidelberg.

[35] Chiu, J. K. H., & Chen, Y. P. P. (2015). Pairwise RNA secondary structure alignment with conserved stem pattern. *Bioinformatics*, 31(24), 3914-3921.

[36] Akutsu, T., de la Higuera, C., & Tamura, T. (2018). A Simple Linear-Time Algorithm for Computing the Centroid and Canonical Form of a Plane Graph and Its Applications. *In Annual Symposium on Combinatorial Pattern Matching* (CPM 2018). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[37] Taufer, M., Licon, A., Araiza, R., Mireles, D., Van Batenburg, F. H. D., Gultyaev, A. P., & Leung, M. Y. (2009). PseudoBase++: an extension of PseudoBase for easy searching, formatting and visualization of pseudoknots. *Nucleic Acids Research*, 37(suppl_1), D127-D135.

[38] Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., ... & Sherlock, G. (2000). Gene ontology: tool for the unification of biology. *Nature Genetics*, 25(1), 25-29.

[39] Gene Ontology Consortium. (2019). The gene ontology resource: 20 years and still GOing strong. *Nucleic Acids Research*, 47(D1), D330-D338.

[40] Byun, Y., & Han, K. (2009). PseudoViewer3: generating planar drawings of large-scale RNA structures with pseudoknots. *Bioinformatics*, 25(11), 1435-1437.

[41] Blin, G., Denise, A., Dulucq, S., Herrbach, C., & Touzet, H. (2008). Alignments of RNA structures. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 7(2), 309-322.

[42] Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., ... & Sherlock, G. (2000). Gene ontology: tool for the unification of biology. *Nature Genetics*, 25(1), 25-29.

[43] Byun, Y., & Han, K. (2009). PseudoViewer3: generating planar drawings of large-scale RNA structures with pseudoknots. *Bioinformatics*, 25(11), 1435-1437.

[44] Schlicker, A., Domingues, F. S., Rahnenf̀uhrer, J., & Lengauer, T. (2006). A new measure for functional similarity of gene products based on Gene Ontology. *BMC Bioinformatics*, 7(1), 1-16.

[45] Roncaglia, P., Martone, M. E., Hill, D. P., Berardini, T. Z., Foulger, R. E., Imam, F. T., ... & Lomax, J. (2013). The Gene Ontology (GO) cellular component ontology: integration with SAO (Subcellular Anatomy Ontology) and other recent developments. *Journal of Biomedical Semantics*, 4(1), 1-11.

[46] Madeira, F., Park, Y. M., Lee, J., Buso, N., Gur, T., Madhusoodanan, N., ... & Lopez, R. (2019). The EMBL-EBI search and sequence analysis tools APIs in 2019. *Nucleic Acids Research*, 47(W1), W636-W641.

[47] Kalvari, I., Argasinska, J., Quinones-Olvera, N., Nawrocki, E. P., Rivas, E., Eddy, S. R., ... & Petrov, A. I. (2018). Rfam 13.0: shifting to a genome-centric resource for non-coding RNA families. *Nucleic Acids Research*, 46(D1), D335-D342.

[48] Pawlik, M., & Augsten, N. (2011). RTED: a robust algorithm for the tree edit distance. *arXiv preprint* arXiv:1201.0230.

[49] Zhang, K., Statman, R., & Shasha, D. (1992). On the editing distance between unordered labeled trees. *Information Processing Letters*, 42(3), 133-139.

[50] Cohen, P. (2002). The origins of protein phosphorylation. *Nature Cell Biology*, 4(5), E127-E130.

[51] Yao, C. H., Wang, R., Wang, Y., Kung, C. P., Weber, J. D., & Patti, G. J. (2019). Mitochondrial fusion supports increased oxidative phosphorylation during cell proliferation. *Elife*, 8, e41351.

[52] Van Hoof, D., Muñoz, J., Braam, S. R., Pinkse, M. W., Linding, R., Heck, A. J., ... & Krijgsveld, J. (2009). Phosphorylation dynamics during early differentiation of human embryonic stem cells. *Cell Stem Cell*, 5(2), 214-226.

[53] Ruvolo, P. P., Deng, X., & May, W. S. (2001). Phosphorylation of Bcl2 and regulation of apoptosis. *Leukemia*, 15(4), 515-522.

[54] Bao, W., Yang, B., Li, D., Li, Z., Zhou, Y., & Bao, R. (2019). CM-SENN: computational modification sites with ensemble neural network. *Chemometrics and Intelligent Laboratory Systems*, 185, 65-72.

[55] Bao, W., Huang, D. S., & Chen, Y. H. (2020). MSIT: malonylation sites identification tree. *Current Bioinformatics*, 15(1), 59-67.

[56] Garcia, B. A., Shabanowitz, J., & Hunt, D. F. (2005). Analysis of protein phosphorylation by mass spectrometry. *Methods*, 35(3), 256-264.

[57] Rubin, C. S., & Rosen, O. M. (1975). Protein phosphorylation. *Annual Review of Biochemistry*, 44(1), 831-887.

[58] Stark, M.J. (2004). The Metabolism and Molecular Physiology of Saccharomyces Cerevisiae (ed. Dickinson, J. R.) 284-290

[59] KREBS, E. G. (1972). Current topics in Cellular Regulation (ed. Horecker, B. L. & Stadtman, E. R.) 99-133

[60] Milburn, M. V., Tong, L., DeVos, A. M., Brunger, A., Yamaizumi, Z., Nishimura, S., & Kim, S. H. (1990). Molecular switch for signal transduction: structural differences between active and inactive forms of protooncogenic ras proteins. *Science*, 247(4945), 939-945.

[61] Watkins, N. G., Neglia-Fisher, C. I., Dyer, D. G., Thorpe, S. R., & Baynes, J. W. (1987). Effect of phosphate on the kinetics and specificity of glycation of protein. *Journal of Biological Chemistry*, 262(15), 7207-7212.

[62] Eck, M. J., & Manley, P. W. (2009). The interplay of structural information and functional studies in kinase drug design: insights from BCR-Abl. *Current opinion in cell biology*, 21(2), 288-295.

[63] Elkins, J. M., Fedele, V., Szklarz, M., Azeez, K. R. A., Salah, E., Mikolajczyk, J., ... & Zuercher, W. J. (2016). Comprehensive characterization of the published kinase inhibitor set. *Nature Biotechnology*, 34(1), 95-103.

[64] Schwartz, P. A., & Murray, B. W. (2011). Protein kinase biochemistry and drug discovery. *Bioorganic Chemistry*, 39(5-6), 192-210.

[65] Warmuth, M., Kim, S., Gu, X. J., Xia, G., & Adrià n, F. (2007). Ba/F3 cells and their use in kinase drug discovery. *Current Opinion in Oncology*, 19(1), 55-60.

[66] Wilks, A. F. (2008). The JAK kinases: not just another kinase drug discovery target. *In Seminars in Cell & Developmental Biology*, 19(4), pp. 319-328.

[67] Fu, Z., Smith, P. C., Zhang, L., Rubin, M. A., Dunn, R. L., Yao, Z., & Keller, E. T. (2003). Effects of raf kinase inhibitor protein expression

on suppression of prostate cancer metastasis. *Journal of the National Cancer Institute*, 95(12), 878-889.

[68] Lin, G., Li, D., De Carvalho, L. P. S., Deng, H., Tao, H., Vogt, G., ... & Nathan, C. (2009). Inhibitors selective for mycobacterial versus human proteasomes. *Nature*, 461(7264), 621-626.

[69] Gross, S., Rahal, R., Stransky, N., Lengauer, C., & Hoeflich, K. P. (2015). Targeting cancer with kinase inhibitors. *The Journal of Clinical Investigation*, 125(5), 1780-1789.

[70] Ali, G. S., & Reddy, A. S. (2006). ATP, phosphorylation and transcription regulate the mobility of plant splicing factors. *Journal of Cell Science*, 119(17), 3527-3538.

[71] Barbour, R. L., Ribaudo, J., & Chan, S. H. (1984). Effect of creatine kinase activity on mitochondrial ADP/ATP transport. Evidence for a functional interaction. *Journal of Biological Chemistry*, 259(13), 8246-8251.

[72] Ben-David, Y., Letwin, K., Tannock, L., Bernstein, A., & Pawson, T. (1991). A mammalian protein kinase with potential for serine/threonine and tyrosine phosphorylation is related to cell cycle regulators. *The EMBO Journal*, 10(2), 317-325.

[73] Shi, L., Potts, M., & Kennelly, P. J. (1998). The serine, threonine, and/or tyrosine-specific protein kinases and protein phosphatases of prokaryotic organisms: a family portrait. *FEMS Microbiology Reviews*, 22(4), 229-253.

[74] LeDeaux, J. R., & Grossman, A. D. (1995). Isolation and characterization of kinC, a gene that encodes a sensor kinase homologous to the

sporulation sensor kinases KinA and KinB in Bacillus subtilis. *Journal of Bacteriology*, 177(1), 166-175.

[75] Watillon, B., Kettmann, R., Boxus, P., & Burny, A. (1993). A calcium/calmodulin-binding serine/threonine protein kinase homologous to the mammalian type II calcium/calmodulin-dependent protein kinase is expressed in plant cells. *Plant Physiology*, 101(4), 1381-1384.

[76] Fabian, M. A., Biggs, W. H., Treiber, D. K., Atteridge, C. E., Azimioara, M. D., Benedetti, M. G., ... & Lockhart, D. J. (2005). A small molecule-kinase interaction map for clinical kinase inhibitors. *Nature Biotechnology*, 23(3), 329-336.

[77] Baratier, J., Peris, L., Brocard, J., Gory-Faurè, S., Dufour, F., Bosc, C., ... & Andrieux, A. (2006). Phosphorylation of microtubule-associated protein STOP by calmodulin kinase II. *Journal of Biological Chemistry*, 281(28), 19561-19569.

[78] Lim, J., Ryu, S., Park, K., Choe, Y. J., Ham, J., & Kim, W. Y. (2019). Predicting drug-target interaction using a novel graph neural network with 3D structure-embedded graph representation. *Journal of Chemical Information and Modeling*, 59(9), 3981-3988.

[79] Qin, T., Zhu, Z., Wang, X. S., Xia, J., & Wu, S. (2021). Computational representations of protein-ligand interfaces for structure-based virtual screening. *Expert Opinion on Drug Discovery*, (just-accepted).

[80] Shen, H., Zhang, Y., Zheng, C., Wang, B., & Chen, P. (2021). A cascade graph convolutional network for predicting protein-ligand binding affinity. *International Journal of Molecular Sciences*, 22(8), 4023.

[81] Son, J., & Kim, D. (2021). Development of a graph convolutional neural network model for efficient prediction of protein-ligand binding affinities. *PloS One*, 16(4), e0249404.

[82] Bao, W., Yang, B., & Chen, B. (2021). 2-hydr_Ensemble: Lysine 2-hydroxyisobutyrylation identification with ensemble method. *Chemometrics and Intelligent Laboratory Systems*, 104351.

[83] Hwang, H., Dey, F., Petrey, D., & Honig, B. (2017). Structure-based prediction of ligand-protein interactions on a genome-wide scale. *Proceedings of the National Academy of Sciences*, 114(52), 13685-13690.

[84] Weisfeiler, B., & Leman, A. (1968). The reduction of a graph to canonical form and the algebra which appears therein. *NTI, Series*, 2(9), 12-16.

[85] Zhu, X., Hu, H., Lin, S., & Dai, J. (2019). Deformable convnets v2: More deformable, better results. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 9308-9316).

[86] Desaphy, J., Bret, G., Rognan, D., & Kellenberger, E. (2015). sc-PDB: a 3D-database of ligandable binding sites-10 years on.*Nucleic Acids Research*, 43(D1), D399-D404.

[87] Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., ... & Bourne, P. E. (2000). The protein data bank. *Nucleic Acids Research*, 28(1), 235-242.

[88] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 8026-8037.

[89] Bjarte Mehus Sunde. Early-stopping-pytorch *https://github.com/Bjarten/early-stopping-pytorch.git* (2018).