Understanding RNA

Enes Algul

Doctor of Philosophy

University of York

Computer Science

September 2021

Abstract

Ribonucleic acids (RNAs) are a group of biologically active nucleic acids in the cell. RNAs are intermediaries between DNA and protein, and they exist in all living organisms and are essential for life. Structurally, they are most similar to DNA. The biological tasks of the RNAs are mostly investigated in biology, computational biology, bioinformatics, medical science, and drug discovery. The fundamental problem is that the number of RNA chains in the cells is very large, and each of them is in different shapes and sizes, and the comparison of RNA molecules is a challenging problem in machine learning and computational biology to determine their functions. It is difficult to understand the function of most of the RNA molecules we find in the real world. In this work, we investigate possible solutions to determine RNA functions that could lead to a massive step forward in understanding biological systems.

Graphs are a type of structured data, and graph-based methods have shown promise for other biological compounds such as protein. RNAs can be represented in graph-structured form, and then RNA graph data can be used in learning applications. In this work, we investigate how to extract useful information from the 1/2/3D RNA shapes, encode these piece of information into structured data, and then apply classification methods to determine their functions. The thesis has four main contributions.

The first contribution is to develop a new large RNA dataset that consists of graph-based representations and 3D Point Cloud representations. The RNA dataset includes 3178 RNA chains, and the RNA chains are labelled in 8 classes according to their reported biological functions. The data set aims to provide a platform to investigate RNA functions in the use of classification methods.

The second contribution of the thesis is to introduce a new graph representation of the RNA molecules based on the minimum free energy (MFE) of secondary structure elements of the molecules. The contribution is to encode each structural component of the 2D RNA shapes as an edge and the total MFE on each 2D RNA component as an edge weight. The weights are determined by a labelling process that considers the MFE of the structure and the particular setting within the RNA. The motivation for this encoding is to reduce the size of the graph representation while giving the secondary structure elements an explicit encoding in the graph.

The third contribution of the thesis is to treat 3D RNA strands as 3D curves using geometric three coordinates (x, y, z) information of C3 atom of each nucleobase. Use three coordinate information to represent each RNA curve with square root velocity function (SRVF), arc length, curvature, and min distance to describe a number of possible graph representations and 3D point clouds representations. Armed with RNA graph representations, the state-of-the-art graph kernel methods applied to determine the relative importance of each RNA graph representation. The applied methods are Weisfeiler Lehman and optimal assignment kernel (WL-OA), shortest paths kernel (SP), and all paths and cycle method (APC).

The last contribution is to use geometric deep learning (GDL) methods to determine the type of RNA molecules. Broadly, two different approaches of GDL methods are applied to report the classification results. The first approach analyses the classification performance using the GDL method based on the graph neural networks (GNN). The applied methods are Deep Graph Convolutional Neural Network (DGCNN), Graph Isomorphism Network (GIN), Structure2vec, Graph U-Nets, and LCGNN$_{GIN}$. In the use of GDL methods on GNN, novel graph representation methods also introduced where the node features of novel RNA graph representations consist of multi-dimensional continuous node features. The second approach is to analyse GDL based on 3D Point Cloud. PointNet, PointNet++, and PointConv are applied RNA 3D Point cloud representations to provide classification results.

Declaration

I declare that this thesis is a presentation of original work and I am the sole author. This work has not previously been presented for an award at this, or any other, University. All sources are acknowledged as References.

A note for ethical consideration: In our research project, we are using data which is provided by world-wide organisations. We are not going to use, collect, store, and share personal and private data. So, there are no ethical issues to gather, store and use data sets. But we are considering University of York's Code of Practice and Principles for Good Ethical Governance in the life-time of the project for documentation, testing, coding, sharing and gathering data. Additionally, we are using University of York Research data management policies for data protection. The purpose of this project has been explained with details in the introduction section of the report. Our research will help in improving the current understanding of the RNA by the biologists, drug designers, computer science researchers, and bioinformatics researchers, especially in improving their studies and relevant researchers. Our research is not going to be directly interacting with the public, animal and consumers. We have neither physical tests nor experiments that directly involves animal or human. As such here will not be any direct negative impact on the public and animals. Our research results will help researchers to understand the type and function of the RNA molecules and how to encode RNA molecules as a graph. In general, the ethics of this work is similar to other RNA and DNA research. It is very clear that; there are limited ethical issues of this work. Through the work, we are consider the University of York's research ethics. We are storing both code and the generated archive according to University of York's research guide.

— Enes ALGUL—

Acknowledgements

First of all, I would like to express my most sincere appreciation and gratitude to my advisor, Professor Richard C. Wilson, for his guidance, assistance, detailed feedback, professional support, advice and motivation to complete this doctoral thesis. I am sincerely grateful to have met Professor Richard C. Wilson and received his invaluable guidance, mentorship and direction throughout this interdisciplinary study.

I would like to thank all my friends who supported me during these challenging four years and made this PhD journey more enjoyable and unique.

Special thanks should also go to the Vision, Graphics and Learning Research Group members at the University of York. The different research topics discussed in the group meetings inspired me to think from multiple perspectives.

Finally, I must express my deepest gratitude to my family for their continuous encouragement, motivation, unconditional love, unending support and patience. Without your presence and tremendous support, it would not have been possible to accomplish this work.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

APC	All Path and Cycle Method
DGCNN	Deep Graph Convolutional Neural Network
DNA	Deoxyribonucleic acid
ESA	Elastic Shape Analysis
GIN	Graph Isomorphism Network
LVQ	Learning Vector Quantisation
MFE	Minimum Free Energy
ML	Machine Learning
mRNA	Messenger RNA
PDB	Protein Data Bank
RNA_ABC	RNA Arc length Base position Curvature
RNA_AC	RNA Arc length Curvature
RNA_A	RNA Arc length
RNA_BL	RNA Base label

RNA_C	RNA Curvature
RNA	Ribonucleic acid
rRNA	Ribosomal RNA
s2v	Structure2Vec
SP	Sortest Path Embedding
SRP	Signal Recognition Particle
SRVF	Square Root Velocity Function
tRNA	Transfer RNA
WLOA	Weisfeiler – Lehmen Optimal Assignment Kernel

Chapter 1

Introduction

1.1 Motivation and Problem Description

In 1869, an important macromolecule was discovered in the nucleus of cells and called nuclein by Friedrich Miescher [1, 2], and in 1889 the nuclein was renamed nucleic acid by Richard Altmann [1]. There are two types of nucleic acids called Deoxyribonucleic acid (DNA) and Ribonucleic acid (RNA), which are very crucial in all cells of all living beings. DNA is the book of the universe that contains all genetic codes of all living organisms [3] except some types of viruses such as HIV, MERS, SARS, and COVID [4]. DNA is like a brain in the cell and instructs to manage other biological molecules. These instructions are received by messenger RNA (mRNA) and transported via transfer RNA (tRNA) to protein, ribosome, and other bioactive molecules. Each protein has a unique function, and as a protein builder, a specific RNA function is responsible for building each protein. There is a set of complex interactions between protein, RNA, and DNA [5].

RNAs are well known to play important regulatory and catalytic roles [6, 5]. These roles include transcriptional regulation, RNA splicing, cell division, protein synthesis (rRNA), catalysing chemical reactions to make proteins (mRNA and rRNA), and RNA modification and maturation [6–11]. RNAs also have significant roles in the treatment of diseases, including viral and bacterial infections, and cancer [12, 4]. RNA is therefore crucial to all life and is important to understand its function. The function of the RNA molecules depends on their structures. Many RNA functions are unknown, the function of the RNAs are needed to be discovered. It is also important to predict the functional type of the newly discovered RNA molecules. Understanding and solving the problem of the RNA functions would help to improve our understanding of the biological systems.

The shape of RNA has been utilised in many studies [6, 13, 14] to determine its functional types. It is essential to know the shape of RNA and to make a comparison between these structures. Understanding the function of the RNA molecules from their structure information is a challenging problem. The arising problem is how to extract useful information from RNA structures and make comparisons between these structures. The current methods are based on 1D/ 2D/ 3D shape alignments [15, 16].

Graphs are a widely used data structure for encoding complex systems and structural shapes. Generally, graphs are used to find solutions to problems in a wide range of applications in maths, computational biology, physics, statistics, chemistry, road systems (navigation studies), social sciences, linguistics, computer science, and many other real-world complex structures/systems [17, 18]. For instance, in road systems, graphs are used to map road systems to find optimal transportation solutions, in maths graphs are used for algebraic solutions, in computational biology and biochemistry graphs are used to encode biomolecules to find representations for protein, DNA or RNA comparisons.

The main motivation of this research is to investigate sophisticated graph representation methods to encode RNA structural shapes and to investigate graph classification techniques for determining the function of the RNA molecules. The key question here is, can we make a comparison between a pair of RNA structures by encoding these structures as a graph, using graph kernels and graph-based deep learning methods?

1.2 Learning on Graph Structured Data

Large bio-molecules and chemical compounds have complex structures and the primary challenge is to represent these data in a structured form for learning purposes. A graph (G) is a broad and rich data structure. This data structure, $G = (V, E, l)$, is defined as a set of vertices (V) and their relation edges (E), and an optional attribute, attribute of the vertices or edges or both, labels (l). The bio-molecules and chemical compounds have a set of structural components, these components can be modelled as a graph. For instance, the atoms of a molecule can be represented as vertices and the hydrogen bond between these atoms as edges, and the name of atoms can be represented as their labels. Therefore, it is straightforward to adapt bio-molecules and chemical compounds in learning tasks via representing them in the graph-structured form.

RNAs have complex structures and these structures can be depicted in a graph-structured form. For instance, RNA includes a sequence of nucleobases, and there is a network between these nucleobases. Each nucleobase can be encoded as a vertex and the interaction between them as edges. The learning process on the graph is related to node and edge features. So the property of the nodes and edges in a graph has potential in learning applications.

1.2.1 Learning with Graph Kernels

A graph models a network to represent interactions between objects/systems or their sub-components. The objects or their sub-components are represented as vertices, and the relations between any pair of them are represented as edges. The arising fundamental problem is how to measure pairwise graph similarities using automated methods.

A graph kernel is a function that helps to automatically compute the similarities between pairs of objects and produce a similarity matrix then use machine learning classifiers to classify the similarity matrix according to the objects labels where the objects are represented in the graph-structured form. Graph kernels are flexible methods that allow applying kernel methods to the graph data [18] and allow using a range of algorithms for pattern recognition [19]. Graph classifications are very complex problems [20], and graph kernels bridge the gap between graph-structured data and a large spectrum of the machine learning algorithms [21] called kernel methods such as SVM, kernel regression, or kernel PCA [18]. In this research, we apply state-of-the-art graph kernel methods to RNA graph representations for classification purposes.

1.2.2 Deep Learning with Structured Data

Deep learning methods have been applied to grids or sequences of structured data where the data is represented in the cartesian space and easy to organise. Graph data are dynamic, and they have no a fixed node order and a fixed reference point. It is difficult to apply existing deep learning methods to graph data.

Recently, graph neural network (GNN) methods have been proposed to find solutions for these problems [22–25]. In a GNN network, as an extension to convolutional neural network, a new graph convolutional layer is proposed to add the network architecture to extract local features [26]. The convolutional layer is a new trainable layer. A convolutional layer is described as follows,

$$Z = \sigma(A'FW + b) \tag{1.1}$$

where Z is the output of the layer, σ is an activation function such as ReLU, A' is a normalised adjacency matrix, F is the feature of the nodes, W is weight, and b is the bias. Therefore, the output layer can be the input layer of a convolution layer then the straightforward neural network models are used to complete network architecture. Convolutional layers in GNN networks mostly have a similar approach with a message passing framework [26].

In the structured data, deep learning methods are applied to solve edge/link prediction [26], node [27] and graph classification [22, 24, 25, 28, 29], and community detection [30] problems. In node classification problems, the neighbour of the nodes is used to find the attribute of the node features in a graph. This method is important for social network analysis to detect if a user is real or fake in a network [27]. Edge prediction is to predict whether node pairs are likely to have a connection in network-structured graph data. In community detection problems, deep learning methods are used to discover a cluster of nodes or segmented structures in a graph [30]. In graph classification problems, GNN methods are used to classify graph data according to their labels.

A wide range of graph neural network methods have been proposed for classification problems in bioinformatics [22, 24, 25, 28, 29, 31, 32]. In comparison to graph kernels, deep learning methods perform better on the graph structured data [31]. In this research, we analyse the state-of-the-art graph neural network methods on RNA graph representations for classification purposes.

3D point clouds are a geometric data type that consists of unordered point sets. In this research, we also represent RNA molecules into 3D point clouds form and apply state-of-the-art deep learning methods on 3D point clouds [33–35] to RNA 3D point cloud representations. Finally, we compare the performance of the deep learning method on graphs, deep learning on 3D point clouds, and graph kernel methods for RNA classification problems.

1.2.3 Potential Applications of Representation Learning in Bioinformatics

Biochemical data includes rich structural information of biomolecules. These structured biochemical data are represented into graph-structured data to use in learning applications for various tasks. In molecular structures, atoms of the molecules can be represented as nodes and bonds between atom pairs in molecules can be represented as edges. In macromolecules such as RNA, DNA, and proteins, the nucleotide sequences in DNA and RNA, and aminoacid sequences in proteins can be represented in graph-structured form [36, 37]. These graph representations can be used in machine learning applications to find interactions between macromolecules or to determine their biological functions in the cell.

Interactions between biomolecules in the cell are highly related to the networks of genetic elements (DNA, RNA). Understanding the functions of biomolecules is important because genetic elements regulate cells, and their genome-wide networks help to improve disease predictions. With learning-oriented representation methods, genetic elements can be encoded into graph-structured forms, where nodes represent nucleotides and edges represent relations [36]. The discovery of the functions of RNA molecules is useful for the development of therapeutics [38]. Table 1 in [38] provides a list of disease-causing trans-acting mutations related to RNA-dependent functions.

Data on potential drug treatments consists of the activity of many small compounds. The compounds of the drugs can be encoded into graph-structured data where each compound represent nodes, and their interaction represent edges [36]. With this representation, machine learning methods can help to find interactions between drugs and can also help to find potential compounds to improve the prediction of drugs.

In this thesis, our goal is to investigate representation methods for encoding 1D/2D/3D RNA structures into graph-structured data to use in learning applications. The potential application of these representations is to help researchers of related fields to understand the biological function of the RNA molecules. The representation methods investigated in this thesis have the potential for learning that help to more accurately predict RNA function. The representation method described in Chapter 5 can also be directly applied to protein and enzyme structures to determine their biological functions in the cell.

1.3 Thesis Organisation

In this research, we investigate representation methods for encoding 1D/2D/3D RNA structures into graph-structured form. We also investigate how to represent RNA structures into 3D point clouds form. Then we apply the state-of-the-art classification methods from different approaches to RNA representations for categorising RNAs by their type of functions. The following chapters of this thesis are structured as follows:

In the Background and Literature Review Chapter (2); At first, we extensively review the literature to understand RNA, RNA primary (1D), secondary (2D), and tertiary (3D) structure, and existing research on investigating the functional type of RNA molecules using graph-based methods. Second, we review the literature to describe graph, kernel methods, and graph kernel methods. We investigate the state-of-the-art graph kernels methods to apply these methods to RNA representations for classification. Then, we briefly explain machine learning and machine learning classifiers.

In Chapter 3, we introduce a new large graph-based RNA data set (York RNA Graph Data Set). Our data set includes 3178 RNA chains, and each chain is labelled with its 1 out of 8 type functions. The goal of this data set is to provide a platform for the research community to investigate graph-based classification methods to determine the type of RNA molecules. This data set was presented at a conference in France and led to a conference paper [39]. This data set is approximately eight times larger than the SCOR data set. We applied the same graph kernel methods and sequence-based method to both data sets. York RNA Graph Data Set perform better than SCOR on each RNA graph representations from 10% to 20% of accuracy. Here we prove that using the state-of-the-art graph kernel methods and a well-organized large data set is assisted to improve the performance of the categorisation [39, 40].

In Chapter 4, we investigate the 2D structure of the RNA. The purpose of this investigation is to present a new graph representation of the RNA based on the secondary structure and minimum free energy. The new RNA graph representation aims to be more compact than a direct graph representation, while explicitly encoding information about the 2D structure. Here, we encode each secondary structure component of the 2D RNA as an edge (E), the total minimum free energy between the bases on these components as an edge weight (W), and each junction as a vertex. We also investigate the performance of the weighted and unweighted graphs on introduced 2D RNA graph representation. For approximately 24 per cent of the RNA chains in our data set, there is no secondary structure; these chains are straight and have no pairs. They do not fold onto themselves to build 2D structures. For these types of RNA chains, we propose an alternative, compact representation by encoding each three base letters of the straight-chain as an edge. The weights for these edges are determined by clustering to provide a set of edge labels. This aims to reduce the size of the structures while maintaining information about the sequence. With a good representation of simple chains, we can get more information from the RNA. If we have more information, we can categorise RNA molecules at a high rate of accuracy.

In Chapter 5, we introduce new representations using rich structural information from the Geometric Shape of the RNA. At first, we represent each RNA chain as a curve using three coordinates (x, y, z) of the backbone sugar of the $C3$ atom. Then we apply elastic shape analysis (ESA) using square root velocity function (SRVF) to encode RNA 3D shape where `'SRVF'` treat RNA as a 3D curve. We also apply `'arc length'` and `'curvature'` to encode the geometric shape of the RNA molecules as graphs. These two methods outperform existing 3D RNA graph representations in determining their functional types. We also compare the obtained results from our encoding methods and the result obtained from a classic representation, which labels `'base position'` according to their closest neighbour. Our representations also perform better than `'base position'` representation. Finally, we introduce joint representations using `'base position'`, `'arc length'` and `'curvature'`.

In Chapter 6, we investigate geometric deep learning methods on graphs (GDL) to predict the function of the RNA molecules using our introduced graph representations. We apply the state-of-the-art GDL methods to RNA graph representations. The methods are Structure2Vec [24], DGCNN [22], GIN [25], Graph-U-Nets [28], and LCGNN [29]. We also introduce RNA 3D Point cloud representations. Then, we analyse the performance of the state-of-the-art GDL methods based on 3D Point Clouds (PointNet [33], PointNet++ [34], and PointConv [35]) on RNA 3D Point cloud representations.

In Chapter 7, we summarise the whole research and conclude the thesis.

Chapter 2

Background and Literature Review

This chapter will extensively review the literature to investigate related work and describes the basic foundations of RNA structural shapes, a standard sequence alignment algorithm, graphs, graph kernels, machine learning classifiers, and RNA graph representations. The chapter is structured as follows. Section 2.1 describes the basic concepts of RNA and its structural shapes (sequence (1D RNA), topology (2D RNA), and the geometry of shape (3D RNA)). Then, it compares RNA and DNA to explore the differences and similarities between both macro-molecules. Section 2.2 explains a standard sequence alignment algorithm which used in DNA and Protein comparisons. Section 2.3 describes the basic concepts of graphs and explain state-of-the-art graph kernel methods. The section also explains machine learning and machine learning classifiers. Section 2.4 focuses on how to represent RNAs as graphs. Finally, Section 2.5 summarizes the Chapter.

2.1 RNA

A nucleic acid consists of a sequence of nucleotides. There are two macro nucleic acids called deoxyribonucleic acid (DNA) and ribonucleic acid (RNA). As presented in Fig. 2.1, the main components of nucleotide are a 5-carbon sugar (2-deoxyribose, ribose), a phosphate group, and a base (one of four molecules = adenine, guanine, cytosine, uracil/thymine) [41]. Sugar is the backbone of nucleic acid (DNA or RNA). The sugar in DNA is called

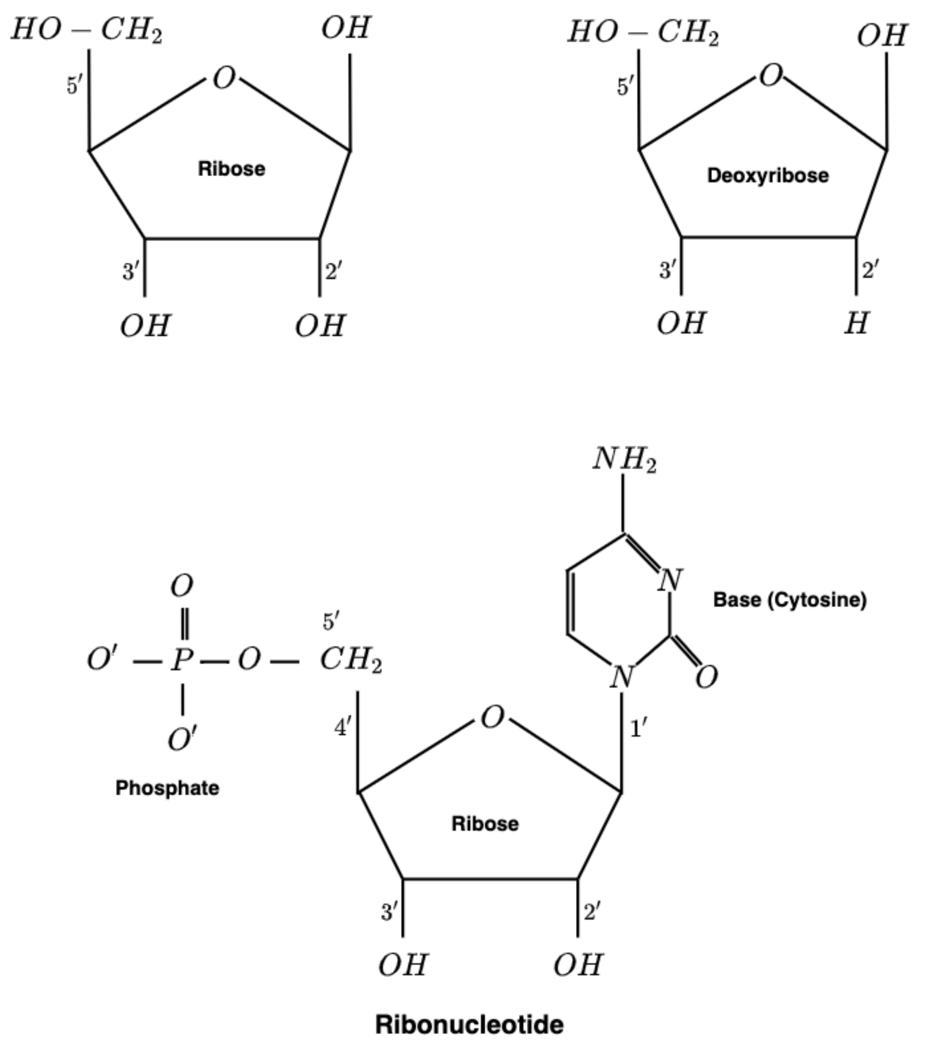


Fig. 2.1 A shape of nucleotide and its components

deoxyribose which has one hydroxyl group $(-OH)$. The sugar in RNA is called ribose which has two hydroxyl group $(-OH)$. Hydroxyl group are attached to the carbon backbone. Each nucleobase of the nucleic acid chains has one backbone of sugar.

Deoxyribonucleic acid (DNA) and ribonucleic acid (RNA) are two highly important biomolecules in the cell. They store and read the genetic information of all living organisms [42]. Both similar biomolecules are nucleic acids and composed of nucleobases. Although the sugar backbone of the polymer is different, and they are very interdependent in their functions. The structural and functional differences between DNA and RNA are as follow,

- DNAs are double-stranded biomolecules while RNAs are mostly single-stranded.(RNA molecules are double-stranded in some certain viruses, and RNA molecules can fold into themselves to build double-helical regions [43].)
- RNA is chemically very similar to DNA in their polymer of nucleotides [17]. Both macro-molecules include bases (nucleotides). The complementary base is different. The RNA bases are adenine (A), guanine (G), cytosine (C) and uracil (U). The DNA bases are adenine (A), guanine (G), cytosine (C) and thymine (T). The base pair of the RNA are (A-T, G-C) while DNA (A-T, G-U). RNA contains ribose and Uracil (U) rather than deoxyribose and Thymine (T).
- RNA nucleotides contain ribose, whereas DNA contains deoxyribose.
- The length of the RNA strands are shorter than DNA strands [44].
- The number of RNA classes is too large in contrast to DNA [45]. DNA molecules are categorising in their number of nucleotide bases (A-DNA, B-DNA, C-DNA, and D-DNA), coiling pattern (right/left-handed DNA), location (chromosomal/cytoplasmic/ promiscuous DNA), structure (linear/circular DNA), repetition of Nucleotide base sequences (repetitive/unique/palindromic DNA), coding and non-coding DNA, and the number of strands (double/single-stranded DNA). Detailed information described in [46]. On the other hand, RNAs have more than 60 functional classes. And each RNA class has sub-classes.
- DNA encodes genetic information in contrast, RNA put this genetic information to use in a wide range of biological functions in the cell.

2.1.1 Primary and Secondary Structure of RNA

The primary structure of the RNA consists of a sequence of base labels. The following RNA sequence is the single strand (strand B) of an Escherichia Coli ribosomal RNA:

"GAUUUGGGGAGUAGCCGAUUUCCGAAAGGAAAUGUACGUGUCAACAUUU UCGUUGAAAAACGUGGCACGUACGGACUGAAGAAAUUCAGUCAGGCGAGACC

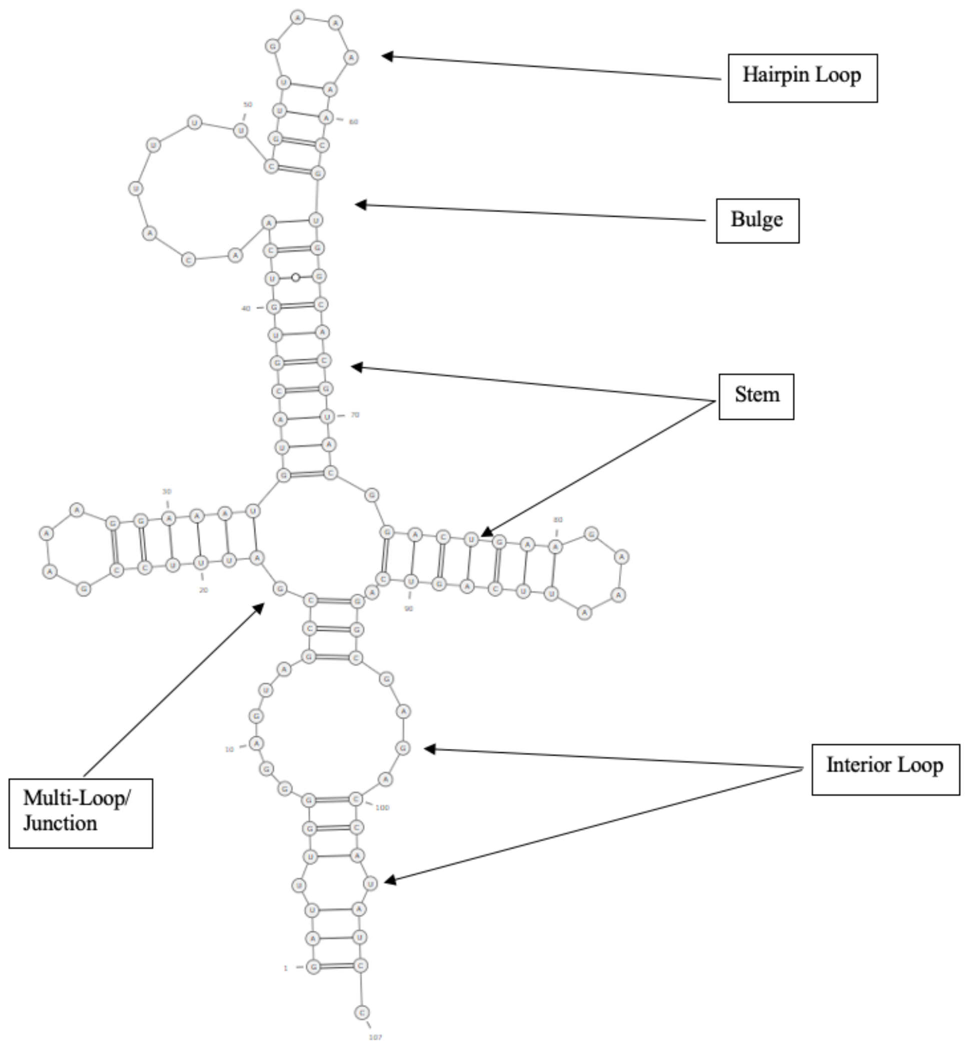


Fig. 2.2 2D RNA structure of the Escherichica coli Riboswitch derived from 4Y1M.pdb (strand B). The 2D RNA structure's topology produced using RNApdbee 2.0 [47].

AUAUCC"

The single strands of RNAs can fold onto itself to build 2D RNA [39] and 3D RNA topologies [48]. As shown in Fig 2.2, the above primary RNA sequence folds onto itself to form a 2D RNA structure. Unlike DNA, RNA is a single-stranded bio-polymer and is encouraged to fold into complex shapes, like proteins, by the matching of base-pairs from the same strand. The secondary structure is formed by both Watson-Crick base pairs (A-U, C-G) [14], wobble base pairs (G-U), and non-standard base pairs. The hydrogen bonds between

A-U, C-G, G-U denote canonical base pairs, while the base pairs between other bases denote non-canonical base pairs [49]. In 2D RNA structures, the canonical base pairs are more stable and more important than non-canonical bases pairs [49].

As presented in Fig. 2.2, 2D RNA structures consist of five structural components. These components are hairpin-loop, multi-loop/junction, bulge, stem/helix, and interior-loop. The uninterrupted base pair portion of RNA molecules are stem/helix, the hairpin-loop is a set of unpaired bases which occurs at the end of some stem sections. The multi-loops occurs between more than two stems, while the interior loops exist between only two stems. The interior loops occur when both sides of the chains are unpaired between two stems. The bulges are similar to the interior loop, and it occurs when only one side of the strand's bases is unpaired between two stems.

2.1.2 Tertiary (3D) Structure of RNA

The existing 3D RNA structures are produced by using 2D RNA topologies and three-dimensional atomic coordinates of the RNA backbone sugar. The 3D RNA structures play significant roles in understanding health and disease in living cells [50]. The function of the RNA molecules is related to their 3D structures. Understanding the 3D structure of RNA is important for exploring its biological functions [51, 11]. According to the Rfam database [14] and 3dRNA v2.0 web server [51], our knowledge of 3D RNA structures is too limited in comparison to their sequence of nucleobases and secondary structures. The 3dRNA generates 3D RNA structures using information from the primary and secondary structure of the RNA. Fig 2.3 shows the 3D structure of the Escherichia Coli ribosomal RNA produced by Matlab molviewer. There are functional and structural differences between 2D and 3D RNA structures. The differences are,

- 3D RNA structures are more complex than 2D RNA structures,
- 3D RNA structures consist of more structural information and demonstrate more molecular details to describe RNA molecules. The 3D RNA structures consist of

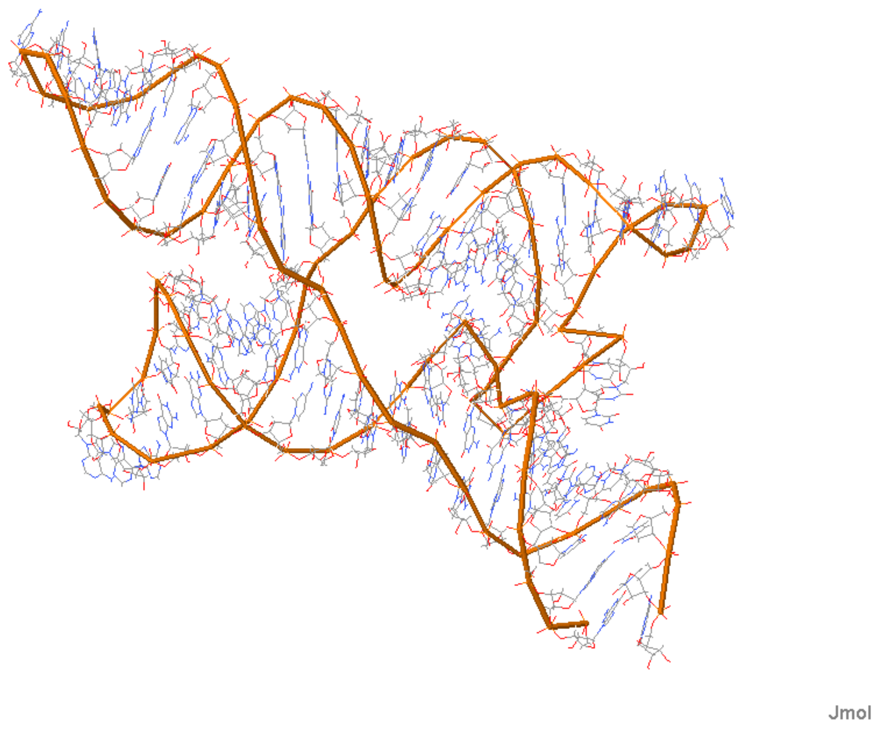


Fig. 2.3 3D RNA structure of the Escherichia coli Riboswitch derived from 4Y1M.pdb (strand B). The 3D RNA structure's image produced using Matlab molviewer.

pseudo-knots and non-Watson-Crick base pairs, whereas 2D RNA structures do not contain this information [14, 52]

2.2 Sequence-based Methods

As a comparison, usually, a standard sequence-based method is employed for DNA or Protein comparison. DNA is encoded by the nucleotide sequences and Protein is encoded by amino acid sequences. The Needleman - Wunsch algorithm [16] is a standard dynamic optimal global alignment method used to align DNA (nucleotide) or protein (amino acid) sequences. The algorithm divides large nucleotide and amino acid sequences into a series of smaller sequences, then uses the solutions of the smaller sequences to find optimal alignments for the large biological sequences. The algorithm produces and assigns a score to all possible alignment and record the highest score, which obtained from the best alignment.

Needleman - Wunsch algorithm is essentially a string edit distance between the strings of base labels. This algorithm also employs the primary structure of the RNA to find similarities between nucleotide sequences. The nucleotides of the RNA is very similar to the DNA sequences and encoded as a string of adenine (A), cytosine (C), guanine (G), and uracil(U). Generally, RNA structures are determined by their sequence of nucleotides. Similar RNA sequences have similar biological tasks in the cell. The strings (A, G, C, U) are aligned using the Needleman - Wunsch algorithm [16]. The distance between two RNA sequences is measured by using the Jukes-Cantor score [53, 54].

$$d = -\frac{3}{4}\log(1 - \frac{4}{3}p). \tag{2.1}$$

Here, p denotes the distance between them, $0 \leq p \leq 1$, the segment of sets that differ $(p = \frac{\text{\# of differences between them}}{\text{length of the sequence}})$.

2.3 Graph Kernel and Machine Learning Classifiers

Graph Theory is a branch of mathematics that involves various areas, including road systems, neurosciences, irrigation networks, chemical processes and structures, computer science, and bioinformatics [55, 17]. Graphs are common data structures that represent relationships between structured data. Graph-based data is becoming more abundant in chemical pathways and protein structures, protein or gene regulation networks, RNA function prediction [40, 39, 56], and social networks [55, 18]. In a structured dataset, graphs model structural components as nodes and the relationships between them as edges. Here, two main problems that arise:

- *"How are the similarities between a pair of graphs measured?"*
- *"How are the similarities between a pair of nodes in a given graph measured?"*

The above two crucial questions are important in the application of RNA classification problems. In computational biology, RNA strands are encoded into graph-structured form. Then, Graph kernel methods use graph representations to determine the functional type of the RNA molecules [40, 39].

Kernel methods have been designed for structural data and perform promising results. Kernel methods are widely used in bioinformatics, such as in Lodhi and Huma [57], where the spectrum kernel, marginalise kernel and fisher kernel was applied for sequence analysis.

Graph kernels are powerful similarity functions used in bioinformatics for pairwise similarity [18, 19]. Graph kernel methods are applied to graph data to generate a kernel matrix. The dimension of the kernel matrix is $n \times n$, where n is the number of graphs in the data set. Then, a machine learning classifier is applied to classify the similarities produced in the kernel matrix. In this way, a graph kernel is a bridge between graph data and a large spectrum of the machine learning algorithms such as SVM kernel regression, kernel PCA [18], KNN and ensemble classifiers (Subspace KNN, Subspace Discriminant, Bagged Trees, and Boosted Trees). Therefore, the rate of graph classification accuracy is also related to both graph kernel methods and machine learning classifiers.

2.3.1 Graphs

Graphs are a type of data structure to represent structural components or objects and their relations. A *graph* $G = (V, E)$ is defined by a finite set of vertices $V = (v_1, v_2, ..., v_n)$ and a set of $E \subset V \times V$ edges [56, 40] where vertices represent structural components/objects and edges represent their relations. An edge exists between adjacent vertices $(v_i, v_j) \in E$ and connects two vertices.

A *labelled graph* $G = (V, E, l)$ is defined by an additional attribute label (l). In a labelled graph, nodes $(l_v : V \to L)$, edges $(l_e : E \to L)$, or both nodes and edges $(l_f : E \cup V \to L)$ can be assigned with a labelling function, therefore, the labelled graph is called as node-labelled, edge-labelled, and fully labelled graph respectively.

A graph can be either directed or undirected. A graph is an *undirected graph* if there is a bidirected edge between a pair of nodes (v_i, v_j). A graph is a *directed graph* if there is an edge directed from node v_i to node v_j but not node v_j to node v_i. If $(v_i, v_j) \in E \Rightarrow (v_j, v_i) \in E$ the graph is undirected otherwise directed.

A *walk* is a sequence of nodes $(v_1, v_2, ..., v_n)$ in a graph such that $(v_i, v_{i+1} \in E)$. If there is walk between every pair of vertices in the graph, then it is called *connected graph.* A walk is a *path*, where there are no repeated nodes in the walk [56]. A graph is *strongly connected* if there is a directed path from any vertex in the graph to any other vertex. A walk is an Euler path if it visits every edge the graph exactly once. A path is a Hamiltonian path if each vertex is visited only once in a graph. If the starting and ending node of the walk is the same $(v_1 = v_k)$ then it is a *cycle*. If there is no cycle in a graph it is called as an *acylic graph.*

Graphs are commonly represented by adjacency matrices. An *adjacency matrix* $A_{i,j}$ is an $(n \times n)$ matrix, where n is the number of vertices in a graph, i is the row number, j is the column number. If there is an edge between node v_i and v_j then $A_{i,j} = 1$, $(v_i, v_j) \in E$. If there is no edge between a pair of vertices $(v_i, v_j) \notin E$ then $A_{i,j} = 0$.

A *weighted graph* is a graph whose edges has weight value $(v_i, v_j) \neq 0$. An *unweighted graph* is a graph whose edges has no weights.

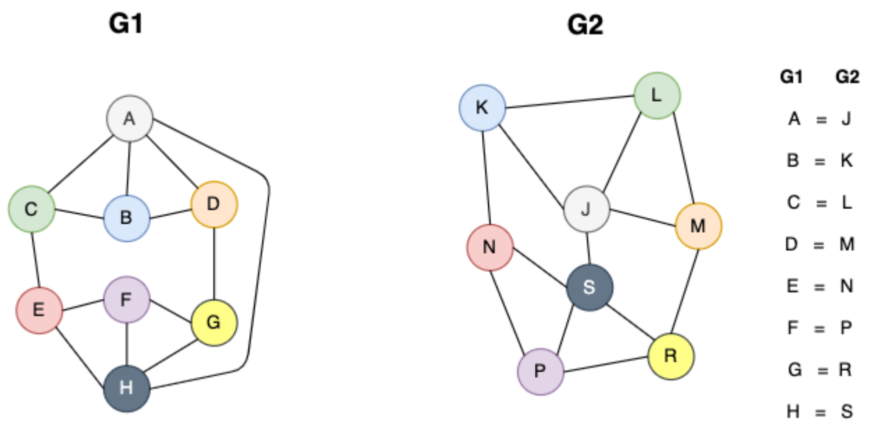


Fig. 2.4 An isomorphism between two undirected graphs ($G1 \simeq G2$)

A graph is also represented with *Laplacian matrix* $L = D - A$. Where $D_{ii} = \sum_j A_{ij}$ ($D \in R^{n \times n}$) is a degree matrix , n is the number of vertices, L is Laplacian, and A is the adjacency matrix. Degree matrix is a diagonal matrix.

Graph isomorphism is a method used to measure similarities between graph pairs. If two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ have the same number of nodes and edges, and the node connectivity via edge is maintained, as presented in Fig. 2.4, there is an isomorphism between them $(G_1 \simeq G_2)$. The topology of both graphs are exactly the same. *Isomorphism* defined with a bijective function $\phi : V(G_1) \mapsto V(G_2)$ where if there is a connectivity between any node pair (u, v) of G_1 and $\phi(u)$ and $\phi(u)$ are connected in G_2.

Graph matching is an important method in computer vision and pattern recognition [58] that is the task of computing the best match for attributed graph pairs. Graph matching is a generalisation of the graph isomorphism problem [58, 59]. Graphs matching can be broadly divided into exact and inexact matching. Exact graph matching requires one-by-one matching of vertices and edges between pairs of graphs. Graph isomorphism is a type of exact graph matching which rarely occurs in pattern recognition. The matching between subparts of graph pairs is called the subgraph isomorphism problem. If two graphs do not have the same topology and the same attributes, then matching problems are defined as inexact graph matching. This is also called error-tolerant graph matching, the graphs topology and their labels do not require to be the same [59]. In this problem, while integrating an error model, it is required to find the best possible match for pairwise graph comparison. Inexact graph matching aims to find similarity scores between graph pairs.

Graph edit distance (GED) is a type of graph matching that measures how similar two graphs are, where algorithms define error-tolerant graph matching (inexact matching) [59, 58]. GED measures the minimum cost of a set of edit operations while transferring a graph into another graph by edge deletion/insertion, node deletion/insertion, and label transformation. The method aims to find the minimum number of steps required to transfer the first graph into the second one for graph matching using insertion, deletion and relabelling of edges and nodes [60]. GED between two graphs defined as

$$GED(G_1, G_2) = \min_{(e_1,\ldots,e_h)\in P(G_1,G_2)} \sum_{i=1}^{h} c(e_i) \tag{2.2}$$

Here $P(G_1, G_2)$ is the set of edit distance paths transforming G_1 to G_2 where each path is a sequence of edit operations. $c(e) \geq 0$ is the cost function associated with each edit operation e_i [58].

2.3.2 Kernel Methods

Kernel methods are popular statistical learning methods in machine learning applications that measure the similarities between two objects [18, 61, 62]. Kernel functions help to classify structural data with a higher rate of accuracy than other related classifiers in real-world applications [6]. A function $k : \chi \times \chi \to R$ is defined as a kernel if there exists a Hilbert space H and some feature map $\Phi : \chi \to H$ such that

$$k(x,x') = \left\langle \Phi(x), \Phi(x') \right\rangle_H \tag{2.3}$$

for all $x, x' \in \chi$ and where $< .,. >$ is the dot product on H, and Φ exists if k is a positive semi-definite and symmetric function [63, 64]. Linear, Polynomial, Gaussian and Dirac (δ) kernels' equation are represented as below;

- $x, x' \in R$ are two n-dimensional vectors,
- α is a non-negative scalar constant (or free parameter), optional
- d is the degree of the polynomials,

$$K_{linear}(x,x') = < x,x' > +\alpha \tag{2.4}$$

$$K_{poly}(x,x') = (< x,x' > +\alpha)^d \tag{2.5}$$

$$K_{gauss}(x,x') = exp(-\frac{||x-x'||^2}{2\sigma^2}), \sigma > 0 \tag{2.6}$$

- for Dirac (δ) kernel x, x' are two points from a set X,

$$K_{delta}(x,x') = \delta_{x,x'} = \begin{cases} 1, \text{ if } x = x' \\ 0, \text{ otherwise} \end{cases} \tag{2.7}$$

A kernel function can use graph data as input for pairwise similarities. A kernel function that consumes pairs of graph data to measure similarity scores between graph pairs is defined as a graph kernel $K(G_i, G_j)$, which satisfies the kernel properties of symmetry and positive semi-definiteness [55, 40]. A graph kernel similarity score values for each pair of graphs in a finite dataset can be formed in a kernel matrix $K_{ij} = K(G_i, G_j)$ (a gram matrix). When the kernel matrix has been produced, the kernel embedding can be used to map the graphs into a feature space representing the kernel. Thus, the kernel embedding projections allows the use of any standard machine learning classifiers directly on the features [40].

A graph kernel should care the following criteria,

- It should be an efficient measure of the similarity for graph pairs [62]
- It should compute in a polynomial time [62], fast to compute

- It should be applicable to all parts of graph [62].

Kernel methods are broadly used in bioinformatics, such as in [57] spectrum kernel, marginalise kernel and fisher kernel have been applied for sequence analysis. Some other types of kernels, which have been used in bioinformatics are subtree kernel, subtree graph kernel, Weisfeiler-Lehmen optimal assignment kernel (WL-OA), and SVM kernel.

The state-of-the-art graph kernel methods (Weisfeiler-Lehmen (WL), WL-OA, Shortest-Path, and All Path Cycle) are explained in following subsections.

2.3.3 Weisfeiler - Lehman Kernel (WL)

The Weisfeiler Lehman method is one of the fundamental methods for graph isomorphism problems. A large number of graph neural networks and graph kernel methods have been inspired by the WL method [18, 65, 22, 25, 24]. The WL method was developed in the 1968 [66], and WL based methods performs better than many state-of-the-art graph classification methods.

The Weisfeiler -Lehman graph kernel is a structural kernel method [18] for graph comparison. The WL algorithm iteratively collects information from the node labels of the neighbouring nodes, produces a multiset label for each node label. The multiset labels consist of a string of the concatenation of original node labels and neighbouring node labels. Then, sort these multiset labels in ascending order. The refined vertices according to their neighbour's node labels are represented in a sorted list. Finally, uses a global hash function to compress these string node labels into a shorter node label to update and relabel the original node labels. Therefore, in each iteration of the graph rewriting process, the node labels are updated and relabelled according to the node labels of neighbouring nodes.

The regular WL graph kernel is defined as follows. $G_n = (V, E, l_n)$ and $G'_n = (V', E', l_n)$ are n^{th} iteration of the graph G and G' rewriting process. V represents a set of vertices and E represents a set of undirected edges. l_n denotes the labelling function and δ refer to Dirac kernel. Then, the Weisfeiler-Lehman kernel can be defined as:

$$K^h_{WL}(G,G') = \sum_{n=0}^{h} k_\delta(G_n, G'_n), \tag{2.8}$$

where,

$$k_\delta((V,E,l_n),(V',E',l_n)) = \sum_{v\in V}\sum_{v'\in V'} \delta(l(v),l'(v')) \tag{2.9}$$

The WL kernel method is one of the most important methods for graph learning. It allows getting efficient experimental results in the term of run time and accuracy with lower complexity.

2.3.4 Optimal Assignment (OA) Kernel

A convolution kernel decomposes the structured data into parts, and take sums over all decomposed parts [67]. Assignment kernels are an alternative function to the convolution kernel that instead of summing decomposed parts, an assignment kernel, finds the optimal bijection between disjoint parts of two objects. In a graph application, this method, first, measures the similarity of vertices in a given graph; second, computes the similarities of vertices in other graphs. Then, the method computes the similarities between the set of graph pairs. This process assures more valid notation for pairwise similarity [67, 68].

Let $k_1 : X \times X \rightarrow R$ be symmetric, positive semi-definite, and non-negative kernel. Then, $x = \{x_1,...,x_{|x|}\}$, and $y = \{y_1,...,y_{|y|}\}$ are substructures of G and G' respectively. $k_1 : G \times G \rightarrow R$, and $k_1 : G' \times G' \rightarrow R$, here, π denote the set of all possible permutations in R. Therefore, the following equation is called as an OA kernel that compares a pair of substructures from both graphs.

$$k_A(G,G') := \begin{cases} \max\limits_{\pi} \sum\limits_{i=1}^{|G|} k_1(x_i, y_{\pi(i)}) & \text{if } |G| \geq |G'| \\ \max\limits_{\pi} \sum\limits_{j=1}^{|G'|} k_1(x_{\pi(j)}, y_j) & \text{otherwise} \end{cases} \tag{2.10}$$

Generally, optimal assignments (OA) is not positive definite [19] and give indefinite functions [67] that are difficult to use in kernel functions. Therefore, there are some arrangements needed before to use OA method.

2.3.5 Weisfeiler - Lehman Optimal Assignment (WL-OA)

The Weisfeiler-Lehmen Optimal Assignment Kernel (WL-OA) is a state-of-the-art method for labelled graph comparison. The WL kernel utilises each graph as a sequence of the graph labels [65]. The optimal assignment method, described in the previous subsection, is used to measure the similarity of the best match between the substructures of the two data points [67]. WL graph kernel, first, counts the common vertex and edge labels then refine them in each of the iterations of the graph rewriting process [65], where the same vertices said to have the same vertex labels. If the same vertices have different neighbours then the vertex labels updated with a refined new label. To update the vertex labels at each iteration of vertex label refinement process, the neighbouring vertex labels are gathered and sorted, where the gathered vertex labels are multi sets, then a global hash function is used for compression of sorted multi sets to construct new and shorter vertex labels [18, 67]. For h iteration, the WL-OA kernel defined as;

$$k(v,v') = \sum_{i=0}^{h} k_\delta(\tau_i(v), \tau_i(v')) \tag{2.11}$$

where, h is the number of iteration, k_δ is a dirac kernel, τ is the sequence of refined vertices { $\tau = (\tau_1, \tau_2, ..., \tau_h)$}

2.3.6 Shortest Path Embedding/Kernel

A walk-based kernel counts the number of similar walks in given two graphs (G_1, G_2) to measure the similarities between them. A walk is also a path if there are no repeated edges. Shortest Path (SP) kernel [62, 69], also called shortest path embedding, is a special type of the walk kernel. The SP method, first, computes the length of all shortest distances between any pair of nodes of input graphs (G_1, G_2). Then, transfer the original graphs to SP graphs

$(SP(G_1), SP(G_2))$ to compare pairs of shortest paths according to their lengths and labels of end point nodes [55]. *SP* graph consists of the set of all possible shortest path of input graphs (G_1, G_2). The *SP* graph preserves the set of vertices of the original graph. Unlike the original graph, there exists an edge between any pair of nodes in the transformed graph. Thus, the number of edges in the SP graph is n^2. Every edge in the shortest path graph that exists between a pair of nodes is labelled by the shortest distance between these two nodes.

Any algorithm that computes the all-pairs shortest path in a graph can be applied to find shortest distance edge labels. Two popular algorithms that solve this problem are Dijkstra, and Floyd-Warshall. Both methods count all shortest path for two graphs (G_1, G_2). The computation complexity of this method is in polynomial time $O(n^3)$. SP method is computationally expensive but still leads to the problems in a high rate of accuracy in graph applications. The equation of the *SP* kernel on graph G_1 and graph G_2 defined as:

$$K_{SP}(G_1, G_2) = \sum_{p_i \in SP(G_1)} \sum_{p_j \in SP(G_2)} K_B(p_i, p_j) \tag{2.12}$$

Here, $SP(.)$ is a set of the shortest path, $K_B(,)$ is a base kernel (base kernel is delta kernel) to compares two lengths where $p_i \leftarrow d(v_i, v_j)$ is the shortest path between node v_i and node v_j, and $p_j \leftarrow d(v_k, v_l)$ is the shortest path between node v_k and node v_l. Therefore, $K_B(p_i, p_j) \leftarrow k_{length}(d(v_i, v_j), d(v_k, v_l))$ compares the length of the all shortest distances in graph $SP(G_1)$ and graph $SP(G_2)$.

2.3.7 All Paths and Cycles Embedding

The all-paths graph kernel [70], recently proposed that based on shortest path kernel, is counts the all possible paths and simple cycles in a graph rather than SPK that counts the shortest path in a graph. A simple cycle is a cycle that in a sequence of nodes the starting node and the ending node are same $(v_1, v_2, ..., v_n, v_1)$ and other nodes are not repeated in a Graph.

$$K_{APC}(G,H) = \sum_{p_i \in PC(G)} \sum_{p_j \in PC(H)} K_B(p_i, p_j) \tag{2.13}$$

where, PC(G) denotes a set of all paths and simple cycle on G and $K_B(.)$ is a base kernel for paths. This algorithm employs graph data where node labels are limited to a maximum of 3 distinct node labels.

2.3.8 Machine Learning

Machine Learning is a part of Artificial Intelligence that uses data to train machines/computers to give them the ability to learn. The learning is coming from the experiences. The general aim is to improve predictions or decisions for future scenarios automatically [71] by the learning algorithms. Machine Learning algorithms are highly important in a wide variety of disciplines, including statistics, natural language processing (NLP), medicine (drug discovery), information theory, cheminformatics, bioinformatics, control theory, neuroscience, self-autonomous systems, computer vision, and computational biology [72–75].

A modern and a good definition of machine learning from the Tom Mitchell [76] based on three features is explained as follow.

Definition 2.1. *"A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improved with experience E." [76]*

Here, we explain E, P and T.

- E: From the experience computer gets a data set as an input. If each file in the data set is labelled then it is a supervised learning algorithm otherwise it is an unsupervised learning algorithm. The source of the experience is very important for the task. The complexity of the input data affects the performance of the given task. Therefore, the data set should be chosen correctly and should be well-represented because lack of knowledge and bad prepared data is hard to be analysed by the ML algorithms.

- T: To learn from the experiences, a task needed to be given. The task can be image classification, speech recognition, link prediction, anomaly detection or other specific aims. A well-defined task helps to solve the problem with better performance.
- P: Then the computer takes the given task and measures the performance (P) from the experiences, for example, it measures the percentage of correctly classified images for image classification; for a robot driving learning problem, it measures the average distance travelled before an error; for a game application, it measures the percentage of games won against the opponent [76].

In bioinformatics applications, a wide range of deep learning and machine learning algorithms have been used such as convolutional neural networks (CNNs), artificial neural network (ANN), recurrent neural network (RNN), support vector machine (SVM), hidden Markov models (HMM), Bayesian network and random forest (RF) [77–81]. For instance, a convolutional neural network for coexpression (CNNC) [82] improved to learn the interaction between genes, their functional assignments, and gene-disease predictions from single-cell expression data. Support vector machine (SVM) based on statistical learning [83] which is one of the supervised methods for binary classification [17], has been applied for the functional similarity of protein in [57], and used in microarray gene expression analysis [84] and computational biology with high accuracy [85]. SVM also have been used in a range of studies such as image segmentation, biology, and text classification [83].

Machine Learning Classifiers

Graph kernel methods are applied to graph data to generate similarities between graph pairs. The result of pairwise similarities is populated in a kernel matrix. To classify the kernel matrix, a machine learning classifier is required to categorize data into a set of classes. The goal is to predict the label of the input data. Here, some important machine learning classifiers are briefly mentioned.

The k-nearest neighbours (k-NN) classification method finds the distances between each test data with k closest instances in training data, then assign a label for each test data based on the majority class in its neighbourhoods [71]. Support vector machines (SVMs) are supervised machine learning classifiers that find an optimal hyperplane with the maximum margin in n-dimensional (n: number of features) space to separate data points for classification [71]. Decision Trees are algorithms that utilise a model in the form of tree structures for classification problems. Each time it uses conditional rule sets to classify data into smaller categories. Ensemble classifiers are machine learning algorithms that use multiple classification models for one classification problem. The prediction results from multiple classifiers are obtained and combined to improve predictive performance. The basic ensemble techniques are weighted averaging, max voting, and averaging [86]. Discriminant analysis [87] are supervised machine learning classifiers that focus on separability among known classes. Discriminant analysis fits a classification function (the discriminant) to data such that the value of the function predicts the class.

2.4 Graph Representations Applied to RNA

RNA molecules are represented into graph-structured forms on two levels. As presented in Fig 2.5, at the first level RNA can be given a straightforward representation as a graph as follows: Each RNA base represents vertices and bonds between RNA bases represent edges [17]. For primary structures of the RNA, the nodes are connected in sequence to generate a graph path. To add the 2D RNA structure, a tool (Ex: X3DNA [88]) can generate edges between nodes using the bound base pairs where graphs are undirected (edges are bidirectional). The number of nucleobases in an RNA chain is the size of the RNA graph representation. The nodes can be labelled with the associated base. This RNA graph representation has been employed in many works such as; in the late 1970s, the secondary structure of the RNA was encoded as a graph by Waterman [89] where nucleotides represent nodes.

On the second level (higher level), 2D RNA topologies are represented as graphs. As presented in Fig 2.6, each structural motifs of the RNA secondary structure can be represented as graph components (edges/vertices). For a tree graph representation, the number of vertices

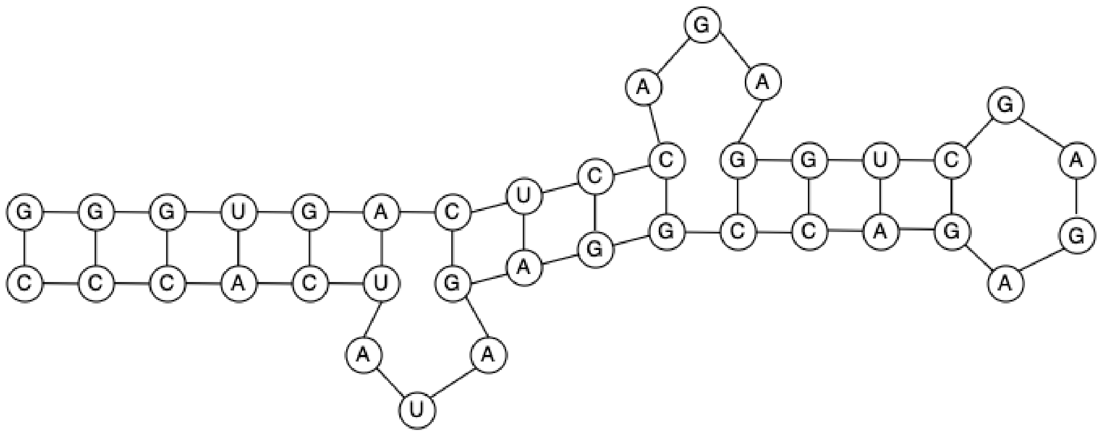
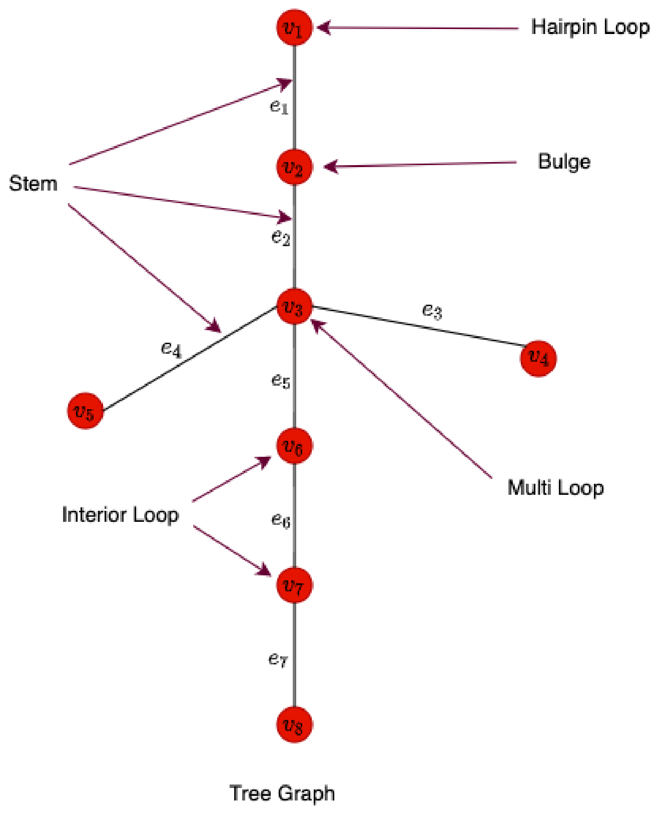


Fig. 2.5 The first level of RNA graph representation. The nucleotides represent the vertices while the bond between nucleotides represents the edges. The 2D structure and seqeunce topology is produced by using RNApdbee 2.0 [47] The sequence and secondary structure of TRNA derived from 1A9L.pdb (strand A). The sequence of the RNA is: GGGUGACUCCA-GAGGUCGAGAGACCGGAGAUAUCACCC

(red dot) represents the number of the unpaired (hairpin loops, bulges, internal loop and multi-loop) structural components of RNA, the number of edges (black lines) represents the number of paired (stem) components of RNAs.

In 2004, RAG [90] generated the 2D RNA structures as graphs including existing and hypothetical motifs. The RAG web resource ranks all vertices of RNA graph representations by their 2D complexity. RAG uses 2D RNA topologies to classify RNA molecules according to their type of functions. RAG database represents RNAs as tree graphs up to 10 vertices (for hypothetical RNA tree shapes) and dual graphs up to 9 vertices (for general RNA structures contain pseudoknots and trees) [91]. They also use directed edges in the graphs with their labelled vertices. As presented in Fig. 2.6, for a tree graph representation, RAG encodes each stem as an edge and each loop (junction, bulge, hairpin loop, interior loop) as a vertex. For dual graph representation, RAG encodes each stem as a node and each loop as an edge. In 2012, Knisley et al [92] extended RAG's dual graph representation that can represent all types of RNA secondary structures. They utilized a multigraph representation of the 2D RNA molecule to classify all possible RNA topologies using existing graph-theoretic descriptors. In 2016, Huang et al. [6] represented each RNA as a graph that can be converted



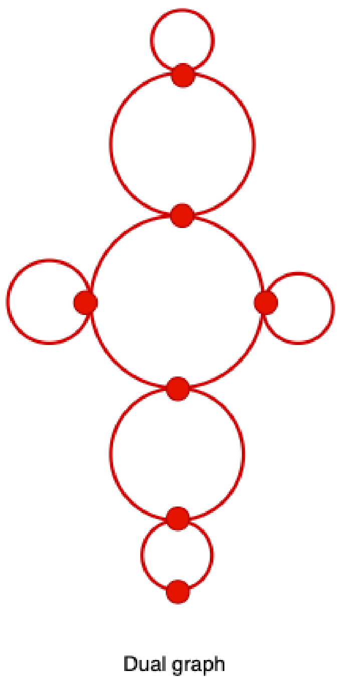


Fig. 2.6 2D RNA topological graph representations of Escherichia Coli ribosomal RNA, derived from 4Y1M.pdb. For tree graph, unpaired sections represent vertices and paired sections represent edges. For dual graph, unpaired sections represent edges and paired sections represent vertices.

to topological spectrums. They described each subgraph of the RNA as its topological fingerprints. They compared topological fingerprints to classify functionally related RNAs.

2.5 Conclusion

This chapter reviewed the literature relevant to the rest of the PhD thesis. This research aims to investigate representation methods and assess these representations in graph-based learning applications. There are very few studies that investigate representation methods for encoding RNA structures into graph-structured form. Understanding how to express 1D/2D/3D RNA structure information is critical to encoding it into a structured data format for use in representation learning.

Recent advances are mostly based on RNA sequences. The primary structure of RNA folds back on itself to form secondary structures and tertiary structures. However, the tertiary and secondary RNA structures consist of more structural information of the RNA molecules. We need to explore how to encode the structural information of 2D/3D RNA structures for use in learning applications for RNA classification problems. Because, in the reviewed literature, the functions of the RNAs are related to their structural features. A good representation of these structural features of the RNAs may improve classification performance more accurately.

In this chapter, we reviewed the literature to describe the primary, secondary, and tertiary structures of the RNA molecules. We also compared DNA and RNA to explore the similarities and differences between both macro-molecules.

In section 2.2, we described a standard global alignment algorithm (Needleman - Wunsch) that is mostly used for DNA and protein comparisons. We will use this method to make a comparison between RNA sequences for multiclass RNA classification problems. This method is based on the string edit distance. We will also compare this method with graph kernel methods on the primary structure of the RNA representation.

In the reviewed literature, the graph representation methods for 2D RNA structures have some limitations. For instance, as described in section 2.4, the RAG database can encode 2D RNA structures for tree graph representations up to 10 vertices and dual graphs representations up to 9 vertices (approx. 240 nucleotide length). The RAG dataset neglects small RNA sequences and cannot represent large RNA sequences in graph-structured forms. Later in this thesis, we will describe graph representation methods that can apply RNA structures of any size. We will also explore how these representations can be used as input graph features in learning applications to determine their functions. The representation methods aim to infer the function of the RNA molecules from their 1D/2D/3D structures.

In section 2.3, we described graphs, state-of-the-art graph kernel methods, and machine learning classifiers. In the next chapters, we will apply these graph kernel methods to RNA graph representations.

Chapter 3

Data

In the previous chapter, we discussed the primary, secondary, and tertiary structure of the RNA and graph based-methods. In this chapter, first, we discuss existing RNA data sets and their file formats, then we introduce a new large RNA Graph Classification data set. The data set is available for the research community to provide a platform to aid their research into RNA classification (https://www.cs.york.ac.uk/cvpr/post/rna/). The source of the data is available from a variety of organisations, the three largest of them are PDBj [93], NDB [94], and RCSB PDB [95]. This work has been presented at GbR2019, a conference in France, and published by the conference [39].

This chapter is an extended version of our paper [39] and is our first research contribution to the PhD thesis. The chapter is organised as follows. Section 3.1 explains fasta, protein data bank (PDB) file formats, and existing datasets. Section 3.2 describes the preparation of the data set, the functional type of RNA molecules from the data set, the distribution of each class in the data set, and their statistical evaluation. In section 3.3, we discuss the contribution of the experiments where the results obtained from our data set in the application of the state-of-the-art graph kernel methods. We also compare the result obtained from our data set and SCOR data set to test the performance of our dataset. In section 3.4, we give a conclusion and summarise the chapter.

3.1 Related Work

In bioinformatics, there exist large datasets of nucleic acids (DNA, RNA) and protein structures. These datasets have a wide range of the file formats (ex: .pdb, .seq, .fasta, .fastq, .csv, .mseq, .gtf, ...). In the reviewed literature, most of the datasets are in fasta and PDB file formats. The fasta files are in the plain sequence format and include a basic nucleobase sequence of the RNA/DNA or amino acid sequence of the proteins [96, 97].

Sequence alignment is a powerful classic method supported by a large research community in the field [15, 98–100]. This method makes a comparison between a pair of macromolecules using their either local or global regions of the sequences to find the similarities. Two important purposes of using sequence alignment in applications of the RNA/DNA are to find matched genes and to predict their functions [71] from gene similarities. In our experiments, we use sequence alignment method to classify RNA molecules as a baseline method. For checking the efficiency of a novel method for RNA classification, it is important to compare a novel method with a standard sequence alignment method.

Numerous tools have been developed to do sequence alignment using a dataset of fasta files [15, 98, 99]. For example, Basic Local Alignment Search Tool (BLAST) [98] is a software and a dataset that provides sequences of the nucleobases in fasta file formats, and the software is providing local alignment. Fasta files do not provide information about the geometry shape of the macromolecules and their secondary structures. Secondary structure and tertiary structure include more information than the primary structure. With a good representation of the secondary and tertiary RNA structure, RNA molecules can be classified with higher accuracy. To make a comparison between RNA molecules using their secondary and tertiary structures, a more compact file format that should contain rich structural information is needed.

The PDB files are more complex and include more structural information of the nucleic acids and protein. The PDB files archive the data of each atom of the macromolecules, their sequence information, their three-dimensional structure (atomic coordinates), and citations. The PDB files also provide the functional type of the macromolecules, their names, the source organisms. The PDB files are standardised according to the atomic coordinates [101]

KEYWORDS	DEFINITIONS
HEADER	The first line of the PDB file is HEADER. Includes PDB ID, classification and date
TITLE	describe the macromolecules with more details
KEYWDS	Includes the list of keywords that describe the bio-molecules
COMPND	The content of the macromolecules are described such as Mol ID, Chain ID, and Molecule description
SOURCE	Includes the biological source of the bio-molecules

Table 3.1 The keywords and descriptions of the main entry of the PDB files

of the macromolecules and provided by many organisations. The three largest of them are Protein Data Bank Japan (PDBj) [93], Nucleic Acid Database (NDB) [94], and The Research Collaboratory for Structural Bioinformatics Protein Data Bank (RCSB PDB) [95]. The contents of the entry of the PDB files have presented in Table 3.1. PDB files provide rich structural information of the 2D/3D RNA structures. Current datasets, such as ARTS/DARTS [15] are using PDB files to represent the tertiary structure of the RNA molecules. They do not provide PDB files. The SCOR [102] dataset provides PDB files, and the PDB files are labelled into 12 types of RNA classes.

The content of the PDB files relevant to the study of RNA is `HEADER`, `SOURCE`, `ATOM`, `HETATOM`, `TITLE`, and `KEYWDS`. In the PDB files, the `TITLE`, the `HEADER`, and the `KEYWDS` sections provide information about the functional classification of the RNA molecules [103], the `SOURCE` section includes chain information [103]. The `ATOM` provides the atomic coordinate in standard, while the `HETATOM` provides nonstandard residues. Both the `ATOM` and the `HETATOM` consist of (x, y, z) orthogonal coordinates of atoms [95]. More keywords and description of the PDB files are explained at the rcsb.org website. In general, the PDB files are used as a hierarchical classification of the structural models of RNA molecules. The PDB files are useful because of their prepared structure. For example, a structure is the top of the hierarchy, and it consists of models, a model consists of a list of chains, a chain consists of residues, and a residue consists of atoms [95].

The discovery of the function of various types of RNA is increasing rapidly. This amount of data is standardised and provided by worldwide organisations. Each chain of the RNA molecules is needed to be labelled and collected standalone in a data set to aid the research community in their investigations. In our research, the standalone RNA chains are required to make a comparison between any pair of the RNA structures.

3.1.1 Existing Data Sets

The current data sets are intended for secondary structure prediction and sequence alignment. In the reviewed literature, numerous of tools and datasets have been proposed to make sequence alignment [98, 104, 15, 99, 105] and to predict secondary structures that inferred from the sequence of the RNA nucleobases. These data sets include only sequence information of the RNA chains where files are in the fasta file formats.

Strand has a curated dataset [104] that consists of the sequence of the RNA nucleotides and their known secondary structure information with known 2D structures collected from public datasets. Strand also provides a tool to investigate, analyse and also download of selected structures. The tool also allows the user to search the type of the RNA molecules. The Strand dataset does not provide tertiary structure information and does not include (x,y,z) coordinate of the atoms of RNA chains to investigate the geometry shape of the RNA. Strand use PDB files to generate 2D structures from the tertiary structures using RNAView [106]. The goal of the dataset and the tool is to conduct comparative sequence analysis and to provide structural components for the secondary structure analysis such as size/type of the components/motifs in an RNA chain.

RNAcentral [107] is a comprehensive dataset of non-coding RNAs. RNAcentral provides sequence and secondary structure of the RNA molecules collected from a variety of the dataset. The dataset is in fasta and json file formats.

DART [15] provides 1333 secondary and tertiary structures of the RNA strands. The data set is generated automatically. The tertiary structure is aligned according to the sequence of the phosphate (P) atom. DART is also providing the sequence of the nucleotides in the fasta file formats. DART does not provide PDB files to the public, but they provide their representations.

HD-RNAs [103] provide 3066 RNA molecules into 9 major types of RNA functions in PDB. The PDB files do not include standalone RNA chains. The PDB files received

Level 1	**Level 2**	**Level 3**
molNaturalOccur (387)	Ribosomal RNA(132)	
	Ribozyme(45)	
	telomeraseRNA(2)	
	Genetic Control Element(8)	
	Viral RNA(100)	
	Transfer RNA(67)	49 Classes...
	SnRNA(9)	
	SRP RNA(10)	
	MessengerRNAoriginal(12)	
	tmRNA(2)	
Evolved (SELEX) RNA(31)	Aptamer(31)	
<undefined>(1)	<undefined>(1)	

Table 3.2 The labelled hierarchical classification of the RNAs. The number of each type of RNAs represents in the brackets.

from RCSB [95]. The nucleotide size of the RNA chains in the data set are longer than 9 nucleotides. Each class is classified according to their source organism. HD-RNAs also provide sequences of all RNA chains in fasta file formats with their corresponding functional classes, source organism, and PDB IDs.

Structural Classification of RNA (SCOR)

The SCOR database was compiled by the Klosterman at al, [102] and labelled into 12 types of RNA functions shown in Table 3.2 on Level 2. The database was prepared by using hierarchical clustering and contains 419 RNA structures in the protein data bank (PDB) file format. The coordinate's information extracted from the PDB files. The hierarchical of per RNA produced from more general to more specific labels inspired by SCOR database [102] which the last update was in 2004. The SCOR database classified RNA as structural, functional and tertiary interaction classification groups [108], and it shows properties of directed acyclic graph (DAG) architectures [109].

3.2 York RNA Graph Classification Dataset Preparation

The PDB files consist of more than one chain of information of the RNA molecules. For instance, in the same PDB files, there might be different kinds of chains that belong to a disparate type of RNA or DNA or RNA-Protein interaction. For example, in 1XNR.pdb file, chain A is 16S ribosomal RNA, chain X is anticodon TRNA, chain M is MRNA, and other chains such as chain B, C, D, ..., T, V are 16S ribosomal protein. It is not straightforward to extract single RNA chains from this type of data. The largest classified database of RNA structure (i.e. RNA strands with functional labels) known to us is that of Klosterman et al. [102] which contains 419 RNA strands.

The purpose of preparing a larger data set is to obtain better performance using appropriate machine learning methods. In our data set, if we have more instances of each type of RNA functions we can categorise RNA molecules in higher accuracy. The best of our knowledge, for using the similar quality of two data sets, learning algorithms perform better on the larger data set. The similar quality means that the data sets share common attributes and parameters. As we presented our experiments in Table 3.4 and 3.5, using a larger data set performs better than a smaller one.

RNA Bricks [110] downloaded PDB files from the World Wide Protein Data Bank web site and split each PDB files by their chains. Their dataset is publicly available. We received a data set from the RNA Bricks that consist 7862 standalone RNA chains. We have extracted 3178 RNA chains from this dataset. For each of these molecules, we have investigated the literature to classify them into one of 8 possible RNA classes. The first step is to look at the `MOL-ID` field in the PDB files, which include information about the type of RNA molecules. In some PDB files, the type of the RNA is not very clear from this field. For these, we undertook a more elaborate investigation using information derived from the `HEADER`, the `TITTLE`, the `KEYWDS`, and HD-RNA [103] in order to determine the type of the RNA chains. We then removed any chains where we were still unsure of the type of function. The result is a curated dataset of 3178 RNA molecules with 8 possible class labels. The dataset can be accessed via https://www.cs.york.ac.uk/cvpr/RNA.html.

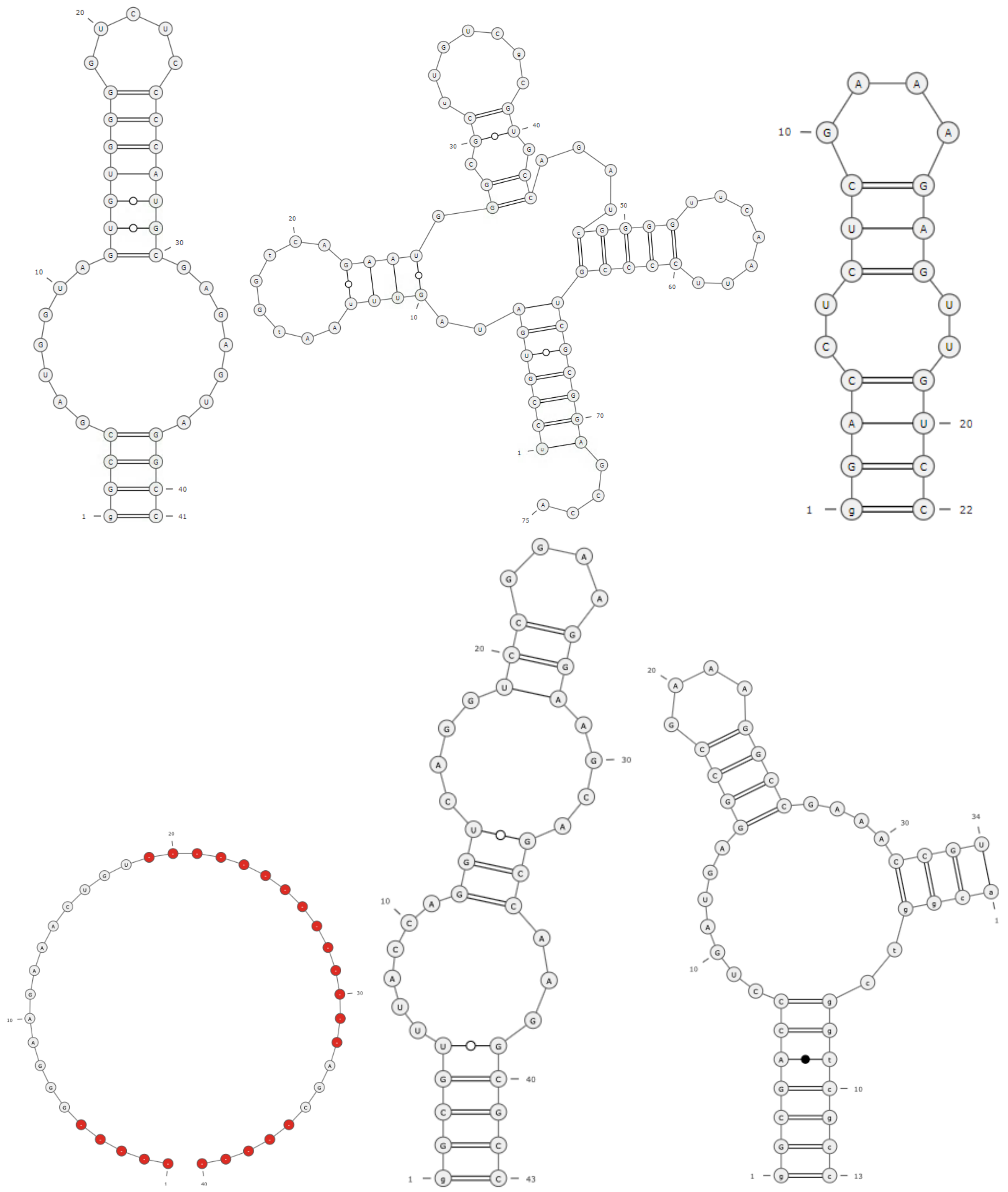


Fig. 3.1 2D RNA structures from left to right: 5S ribosomal RNA (1a51.pdb chain A), transfer RNA (1asy.pdb chain R), messenger RNA (1n66.pdb chain A), riboswitch (1am0.pdb chain A), SRP (1cq5.pdb chain A), and Ribozyme (1hmh.pdb chain C). The 2D structure's images produced using RNApdbee 2.0 [47].

CLASSES	KEYWORDS/ DESCRIPTION
RIBONUCLEASE	RIBONUCLEASE P, RNASE P
RIBOSWITCH	APTAMER
MRNA	UTR, EXON
RIBOZYME	S-TURN, CATALYTIC RNA, HAMMERHEAD, GLMS
RRNA	4.8S, 5S, 5.8S,16S, 18S, 23S, 25S, 26S, 28S, 30S, 40S, 50S, 60S, 70S, 80S, A-SITE OF HUMAN RIBOSOME, A-SITE OF HUMAN MITOCHONDRIAL RIBOSOME, A-SITE OF BACTERIAL RIBOSOME, SARCIN/RICIN 28S RRNA
SRP	4.5S, 7S, 7SL
TRNA	A-site, P-site, tRNA X-MER, FMET, FME, INITIATOR, INI, PRIMER CODON, ANTICODON,ACCEPTOR, tRNA-fragment,
OTHER	viral RNA, miRNA, snoRNA, IRES RNA, and some undefined RNAs such as 5’ RNA, 16-MER, 192-MER, 28-MER, 119-MER, 97-MER etc.

Table 3.3 The labelled classification of the RNAs and their descriptions.

There are different types of RNA molecules, each type has various tasks in the biological process, and their structures differ as presented in Figure 3.1. Each RNA type has numerous sub-classes. We collect RNA molecules under 8 categories according to their functional classes as follows. First, we categorised RNA molecules into 7 major types. The sub-classes of each RNA type also labelled the same as their parent classes. Second, we collect all other types of RNA molecules where the number of RNA classes is too small in a new category. Therefore, our RNA data set classified into 8 categories. We have not labelled our dataset according to the source organism. Selected types of RNA function were inspired by HD-RNAs [103]. HD-RNAs contain 9 major types of RNA classes. The 8 types of RNA functions we selected are the same as the 8 types of RNA functions in HD-RNAs. The RNA classes from Table 3.3.,

- **Messenger RNA (mRNA):** All of the cellular activities in the cell are managed by instructions received from the DNA. mRNA, the most widely familiar types of RNA [17], receives genetic information coded from DNA in the nucleus and export it to the cytoplasm. Then, in the protein, ribosome reads this genetic information from the mRNA and translate them to an amino acid to make protein. So, each mRNA molecule is responsible for building a specific protein [3, 111].

- **Transfer RNA (tRNA):** tRNAs participate in protein synthesis. The length of the nucleotide sequence of this family is from 70 to 100 nucleobases, and this length of the nucleotide is smaller in comparison to other types of RNA chains [112]. tRNA is responsible for transferring correct amino acid to the ribosome during protein synthesis [112]. This process is also called translation.

- **Ribosomal RNA (rRNA):** rRNAs are an essential structural component of the ribosome, and they also have enzymatic activity [17]. rRNA's role in the cell is to translate the genetic code in mRNA, into protein to build protein [113].

- **Ribozyme:** Ribozymes [8] are known as RNA enzymes that act as catalysts. Ribozymes are involving essential biological process in the cell, such as catalyse chemical reactions, replicate RNA genomes. The more fundamental roles of the ribozymes are RNA splicing, translation and degradation [8, 114]. The fourteen types of ribozymes explained in [114].

- **Riboswitch:** riboswitches are small genetic devices [115] in the gene regularity systems and modulate the expression of a gene at the translation and transcription process [116].

- **Ribonuclease:** ribonucleases, the most studied enzymes of the 20^{th} century [117], are a superfamily of enzymes [118] in the RNA-based system that catalyse the RNA degradation, RNA maturation and turnover [119].

- **Signal recognition particle RNA (SRP):** SRP, a part of ribonucleo-protein (protein-RNA complex), involves targeting translocation of membrane proteins and secretory proteins [120].

- **Other:** We labelled all other RNA classes, which the number of classes too small, in the OTHER section.

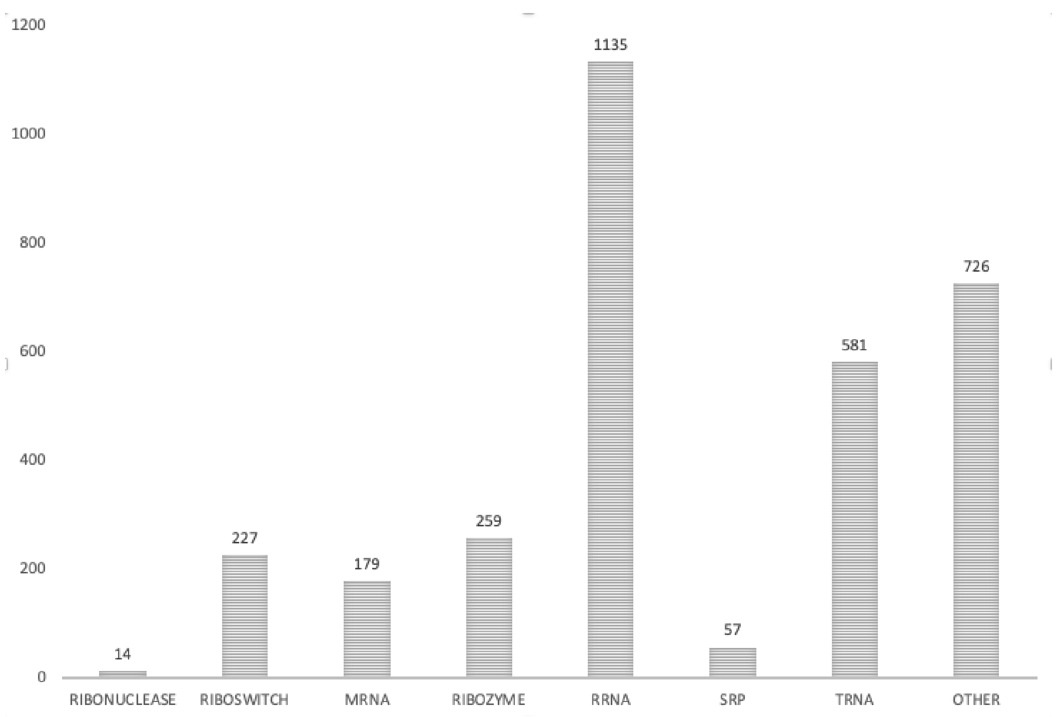


Fig. 3.2 Distribution of the RNA Classes

3.2.1 Distribution of the RNA Classes and Statistic of Nucleotide Sizes

Here, we present the distribution of the RNA types represented in Fig 3.2. The dataset contains 3178 RNA strands into 8 types of categories. The percentage of instances in the dataset is as follows, 35.71% (RRNA), 22.84% (OTHER), 18.28% (TRNA), 8.15% (RIBOZYME), 7.14% (RIBOSWITCH), 5.63% (MRNA), 1.79% (SRP), and 0.44% (RIBONUCLEASE). The results show that the RRNA is the most predominant instances, and the least numerous class is ribonuclease. Overall, we chose the most general types of RNA molecules from various classes and numbers.

In the literature reviewed, some classes of RNA molecules have been studied more than others. As presented in Fig. 3.2, we extracted 3178 RNA strands from the RNA Bricks data set. In our observations, when we labelled them, we found that the number of samples in each RNA class was imbalanced. As shown in Figure 3.2, each RNA class consists of a different number of RNA strands. The number of RNA strands for each class in the dataset ranges from 14 RNA samples (RIBONUCLEASE) to 1135 RNA samples (RRNA).

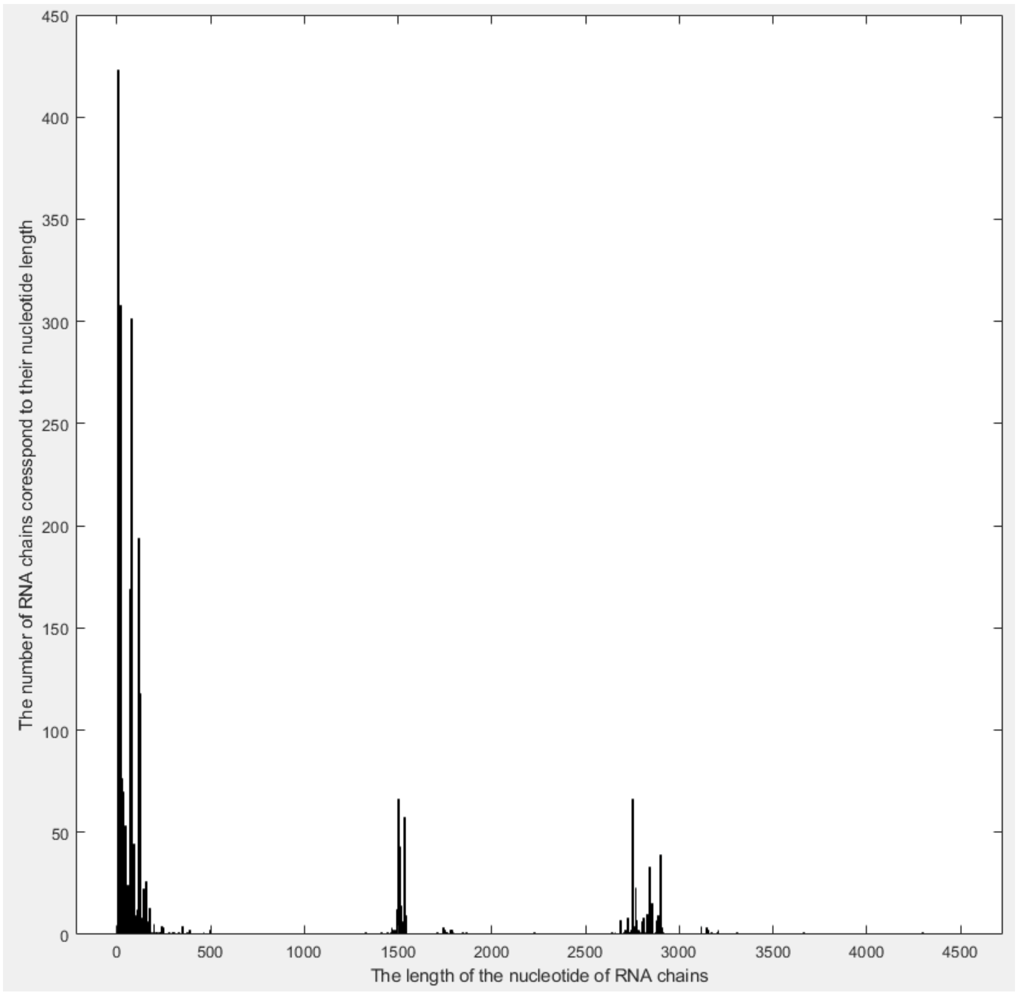


Fig. 3.3 The Histogram of the RNA chains according to their nucleotide size

In our data set, 332 RNA chains' nucleotide lengths are from 6 to 9. The RNA chains in this group are 10,45% of the dataset, and they are unpaired straight chains. This group of the molecule has no topologies. The nucleotide size of the largest group of the dataset is from 10 to 100 bases. This group includes 1798 RNA strands which are approximately 56.57% of the data set. This group of the molecules fold into itself to build 2D topologies. The RNA chains in this group are more similar to DNA. For approximately the nucleotide size of 32.98% of the dataset are from 101 to 4298. The RNA molecules in this group can fold into itself multiple times, and structurally they are more similar to protein structures. More detail of the this group's nucleotide size is as follows, 469 RNA chains nucleotide sizes are from 101 to 500, 277 RNA chain's nucleotide sizes are from 1326 to 1861, 286 RNA chain's nucleotide sizes are from 2227 to 2912, 15 RNA chain's nucleotide sizes are from 3119 to 3662, and nucleotide size of one chain is 4298. The average length of the nucleotide size is 447.106. Fig 3.3 presents the histogram of the nucleotide size of the RNA strands in the dataset.

3.3 Experiments

In this chapter, we introduced a new large graph-based RNA data set. The data set includes 3178 RNA strands, where RNA strands collected under 8 types of RNA functions in PDB file formats. The functional classes and their corresponding keywords/descriptions are presented in Table 3.3. We chose the major types of RNA molecules. Each type of RNA chain has abundant subclasses. For example, ribonuclease has more than 40 subclasses, RRNA has more than 60 subclasses, and other main types of RNA also contain numerous subclasses. To avoid labelling a wide variety of RNA types and their subclasses, we limited the data set into 8 main RNA types. Each PDB files in the data set are manually labelled. The size of our data set is approximately 8 times larger than the SCOR data set.

A purpose of our experiments is to check the validity and reliability of our data set, where we compare the result obtained from our data set to the SCOR data set. The current representations are the baseline for RNA graph representations. We checked if our data set obtains better results than the results obtained from the SCOR data set using the same

	Sequence,	Topology,	Topology + Sequence,	Topology + Geometry,	All
WL-OA	**84.0**	68.7	81.9	62.5	80.1
SP	77.1	72.3	76.8	67.3	**78.3**
APC	**76.1**	56.3	75.7	65.9	...
SA	73.2	...	...	...	...

Table 3.4 Classification accuracies[40] for the SCOR dataset using a variety of methods and representations.

practical representations and comparison methods. As presented in Table 3.4, even satisfying result were not produced by using state-of-the-art methods on SCOR data set. We proved that our data set is valid, and we need a larger data set where each type of RNA functions include more instances.

We performed experiments in a variety of representations on our dataset. The RNA is encoded as a graph using a straightforward representation. In these experiments, our graph representations are unweighted. For all graphs in our data set, we consider the edge weights as 1 for all edges. In all graphs, each vertex represents an RNA residue and is labelled with the base code (A, G, C, U or a rare non-standard base). An adjacency matrix encodes the graph edges, which join all residues in sequence and any base pairs. Base-pairs are detected using the X3DNA program 'find-pairs' [88]. The vertices are labelled with a geometric 'type' label; 1 if it is part of a base pair, 2 if it is unpaired but within 6.5Å of another base, and 3 otherwise. Two sets of 3D coordinates are also provided, firstly for the backbone position of the residue (the position of atom *C*3') and secondly the centroid position of the base. From this data, information about the secondary and tertiary structure can be inferred. These methods are explained in more detail later in the thesis.

As presented in Table 3.5, we experiment that graph-based methods can be used to classify RNA molecules. To evaluate the effect of different types of structure (broadly corresponding to the primary, secondary and tertiary structure), we include information from the topology, sequence, and the geometry of the shape of the RNA. The primary structure is encoded as a graph where the base codes (A, G, C, U or a rare non-standard base) are represented as vertices and their and graph edges exist between adjacent sequences.

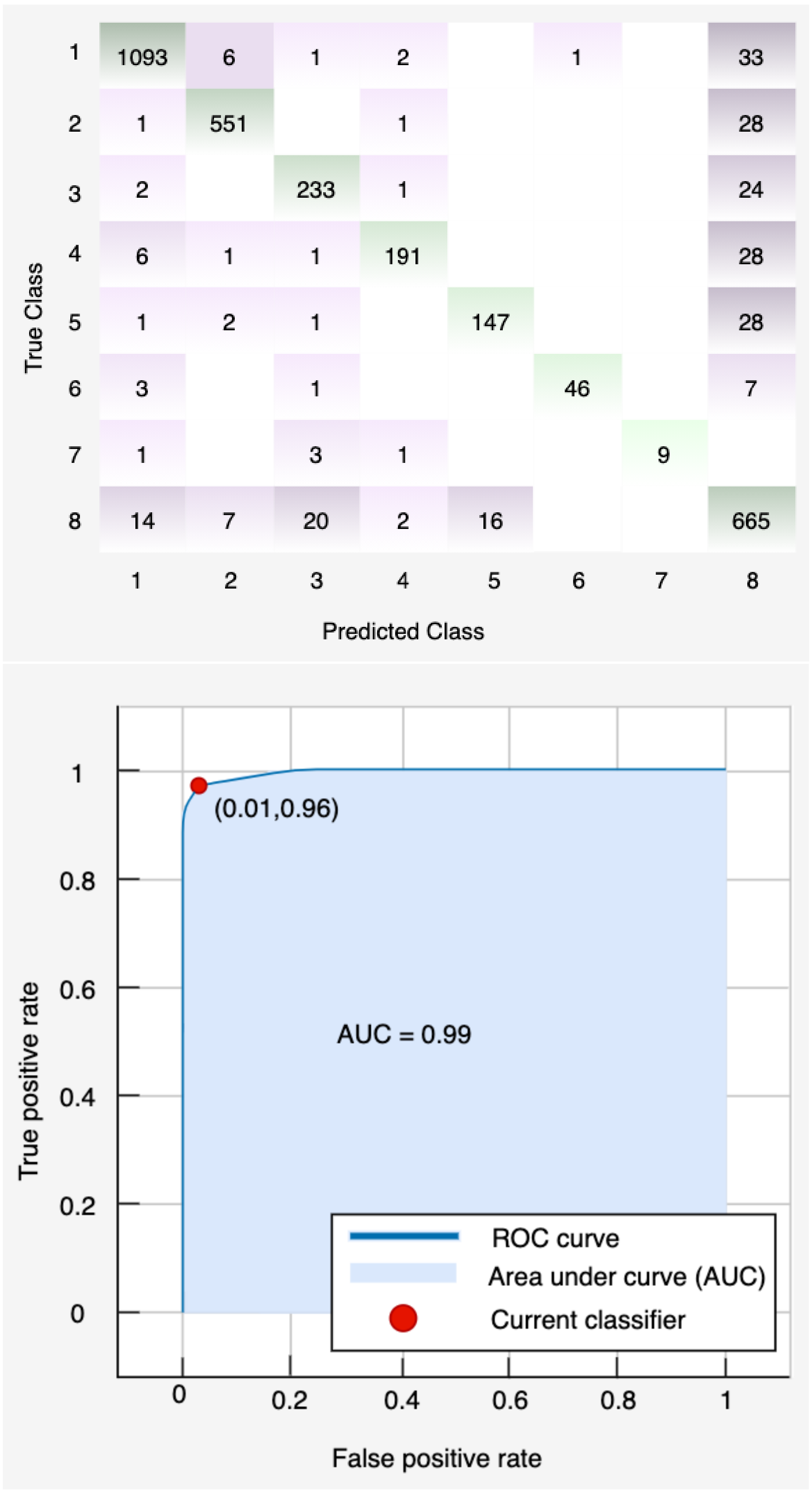
	Seq only,	Top only,	Seq +Top,	Geo + Top,	All
WL-OA	**92.0**	73.1	**92.4**	**87.1**	90.2
SP	91.3	79.5	91.1	86.7	**90.8**
APC	90.3	**85.4**	89.9	85.5	...
SA	89.2	...	...	...	...

Table 3.5 Classification accuracies for the York RNA graph classification data set using Weisfeiler Lehman Optimal Assignment (WL-OA), Shortest Path (SP), All-Paths and Cycle methods (APC), Sequence Alignment (SA), and a variety representations [39]

Therefore, *Seq* includes the graph edges corresponding to the sequence only, and the base labels. To represent secondary structure as a graph, we encode the sequence of nucleotides as vertices, and edges exist between paired bases and adjacent bases. Thus, *Top* includes the graph edges (including the sequence and base-pair edges) but no base labels. For tertiary structure graph representation, we encode geometric type label as vertices, and edges exist between paired bases and adjacent bases. In this manner, *Geo* adds additional labels to the bases corresponding to the geometry type labels described in the previous paragraph. The combinations are the union of these sources of information.

We test graph-based embedding methods (Weisfeiler -Lehman Optimal Assignment method (WL-OA), Shortest Path Kernel (SP), All Path Cycle Kernel (APC)), and a standard sequence alignment (SA) method (Needleman-Wunsch) on our RNA dataset. These methods were explained in the section 2.3. We also tried a variety of machine learning classifiers; Subspace KNN, Subspace Discriminant, Linear discriminant, Boosted Trees, Cosine KNN and Bagged Trees. In our experiments, Subspace KNN outperforms the results with APC methods and SP methods on three sources of information of the RNA. Bagged Trees outperform the best results with the SA method. On the other hand, WL-OA performed its best results from a variety of methods on a variety of representations. Subspace KNN performs the best result on Geometry Shape of the RNA + Topology, and All Label; Subspace Discriminant and Linear Discriminant demonstrate best results on Residue Label, and Residue + Topology Label; Boosted Trees performs the best result on Topology Label.

The experimental results on our RNA dataset represent that, Weisfeiler-Lehman Optimal Assignment (WL-OA) methods outperform the Shortest Path (SP) method, All Path Cycle



True Class \ Predicted Class	1	2	3	4	5	6	7	8
1	1093	6	1	2		1		33
2	1	551		1				28
3	2		233	1				24
4	6	1	1	191				28
5	1	2	1		147			28
6	3		1			46		7
7	1		3	1			9	
8	14	7	20	2	16			665

Fig. 3.4 Confusion Matrix and ROC Curve was obtained from York RNA graph classification data set by applying WL-OA method to the Residue + Topology Label

(APC) method, and Sequence Alignment (SA) method. The accuracy with WL-OA increased up to 92.4 percent, which is the best performance in all methods. The results obtained from our dataset (Table 3.5) [39] largely support the results obtained from SCOR dataset (Table 3.4) [40] in that all graph-based methods outperform sequence alignment even when only the sequence data are available. In observed experiments, the base labels are important to the classification, with a drop-off of between 4–10% when they are not included. The APC kernel can only be evaluated with three labels, which means that it cannot be used in the experiments with rich label sets and explains the lower performance overall.

Finally, the experimental results are obtained from the same state-of-the-art comparison methods and RNA representations to compare the performance of the SCOR dataset and our dataset. York RNA Graph classification dataset performs better from 8% to 16% than SCOR dataset using sequences of the bases, from 5% to 19% using secondary structures, from 11% to 15% using sequence + topology; from 19% to 25% using geometry + topology, and from 10% to 12% using joint representations. The results are shown in Table 3.4 and Table 3.5.

3.4 Summary

In this chapter, we introduce a new large RNA graph classification dataset. The technical details of the preparation of the dataset are explained. The keywords and descriptions describing the types of RNA molecules are taken from a wide variety of articles in the literature. All PDB files in the data set double-checked and manually labelled. Now, if required, the research community can expand this dataset with less effort. The PDB files include information of the standalone RNA chains, their sequence of the bases and (x,y,z) atomic coordinates. We did not add any RNA chains to our dataset where the nucleotide size is less than 6 nucleobases. This type of bases has no secondary structural components.

We also described a variety of graph representation of the RNA structure where the information extracted from our dataset. We applied the state-of-the-art graph kernel methods and a standard sequence-based method to measure the similarity of two RNA molecules. Then, we applied machine learning classifiers to classify RNA molecules into high-level classes. We have shown that graph kernels can be used to classify RNA with high accuracy. As presented in Table 3.5, the best results received on the Residue Label (sequence information) and Topology Label using WL-OA with an accuracy of 92.4%. We refer to this graph representation (Residue plus topology) as RNA_R. The worst results received on the nucleotide sequences using a general Sequence Alignment method with an accuracy of 89.2%. Our results demonstrate that graph-based methods improve on sequence alignment for RNA classification.

We compared the performance of our dataset and SCOR dataset using the same methods (kernel methods and representations). Our dataset performs better on each method. Our work has been accepted by the GBR2019, a conference in France, and also published in LNCS on Springer [39]. In the next two chapters, we investigate more sophisticated descriptions for encoding RNA structures.

Chapter 4

2D Topology

The biological functions of RNA are determined by their structures [121]. The secondary structures are predicted structures. The actual shape may differ than the predicted one. Hence, it is crucial to use a powerful tool that correctly generates the 2D RNA structure. The prediction of the functional tasks is also related to getting useful information from the RNA structure. The secondary structure of the RNA consists of a series of base pairs (helix/stem) and loops (bulge, hairpin, interior loop, and junction). Currently, two methods are mostly used by the research community to generate the secondary structure. The first of these is to use free energy minimisation. The second method is to use the distance between two atoms of nucleotides to decide whether they paired or not.

In this chapter, we introduce a new graph representation for encoding 2D RNA structure. The new representation is based on the minimum free energy found in structural components of the 2D RNA shape. This representation aims to reduce the size of graphs to improve the efficiency of learning methods and at the same time, extract more information from the 2D RNA structures to determine their functional classes. This chapter is our second research contribution to the PhD thesis. The chapter is organised as follows. Section 4.1 explains the problem statement. In section 4.2, we propose solutions for representation problems and introduce our proposed graph representation. In section 4.3, we discuss the contribution of the experiments where results obtained from our proposed 2D RNA representation, generated from our RNA graph classification data set. Finally, in section 4.4, we summarise the chapter.

4.1 Problem Statement

In this section, we will define the representation problem for encoding 2D RNA structure as a graph for use in a learning method for determining the function in cells.

The sequence of the nucleobases is the primary structure of the RNA. The nucleotide sequences fold onto itself to build secondary structures. The structural components/motifs of the secondary RNA structure is presented on the right-hand side of the panel in Fig. 4.1. The secondary structure is the topology of RNA and is more complex than the primary RNA structure. Complex RNA structures include more information than their primary structures. It is straightforward to encode the RNA primary structure as a graph by representing each nucleobase of RNA sequences as vertices and edges between adjacent vertices. Whereas, it is more difficult to encode complex 2D RNA structures as a graph. The following challenges have been faced.

- A first problem is how to encode the 2D RNA structure as a graph.
- A second problem is how to extract useful information from the 2D RNA structure to encode the structure as a graph.
- A third problem is how to reduce the shape of the 2D RNA structure to encode the 2D RNA structure as a graph.
- A fourth problem is how to make a comparison between a pair of RNA strand using their 2D structures to predict their function.
- A fifth problem is how to address the multi-class 2D RNA classification problem according to their reported biological tasks in the cell using their graph representations.

In this chapter, new sophisticated approaches have taken to address these problems. In these approaches, first, each 2D RNA structural components are encoded as edges and the junction between the edges as vertices; second, the total minimum of free energy is encoded as edge weight to extract more information for representing 2D RNA in a graph-structured form. With this approach, the graph is reduced in size, more information is extracted from the 2D RNA structure using the reduced shape, and the 2D RNA structure is successfully encoded as a graph. Finally, the state-of-the-art graph kernel methods are applied to solve the multi-class classification problem by making a comparison between a particular pair of RNA structures to predict whether similar structures share similar functions. In this work, York RNA graph classification data set is considered, which consists of 3178 RNA strands labelled into 8 types of RNA families and is described in Chapter 3. 3178 RNA chains include 1420903 nucleobases, therefore, the average nucleotide size of this data set is 447.106.

The right panel of Fig. 4.1 shows an example of the 2D RNA structure of an RNA molecule, with some typical structural components such as stems, hairpin loops, and bulges. The structural components of the 2D RNA have been predicted by free energy minimisation and represented as a graph in numerous work [122, 89, 123, 124]. First, Tinoco et al. [122] proposed a method to generate 2D RNA structures using minimum free energy (MFE). Tinoco's method was improved by Waterman [89] in 1978. Waterman [89] also encoded the 2D RNA structure as a graph. Third, Zuker and Stiegler [124] proposed a dynamic algorithm in the 1980s to estimate the secondary structure of the RNA using free energy between loops. Their algorithm proposed to predict the 2D structure of the RNA only using nucleobase sequences. Then, the most popular research like Vienna RNA Software package and Mfold Web Server (maintained by Hofacher [125] and Zuker [126] respectively) focused on predicting the 2D RNA structure using the MFE.

Our approach for representing RNA is to use the derived secondary structure to provide graph topology and the associated minimum free energy (MFE) to give more detailed labelling of the structural element. We investigate how similar topologies and MFE help to predict the functional type of the RNA. Following questions below help to address the problems.

- **Q1** Is the form of similar structures indicative of similar activities in the biological process?
- **Q2** Is it useful to predict the type of RNA using the 2D RNA structure and MFE?
- **Q3** How can the minimum free energy be encoded as a graph?

- **Q4** How weighted graph affects the performance of the classification accuracy?

Consequently, representing each 2D RNA structural component as an edge and the total MFE in each 2D RNA component as an edge weight can reduce the size of each graph in the dataset and additionally represent more information. The reduced 2D RNA graph representations can increase the training speed due to the small size of the graphs, and reduced 2D RNA graph representation can assist classify RNA molecules more accurately as it contains both 2D RNA structural information and MFE.

4.2 Solution

In the reviewed literature, the secondary structures of the RNA are highly related to their structural components. Numerous methods have been proposed to predict secondary RNA structure [7, 10, 121, 127–130]. As aforementioned in this chapter, all secondary RNA structures are predicted structures. In the reviewed articles in the literature, various approaches have been proposed by the research community to construct secondary RNA structures.

X3DNA [88] is a tool that helps to generate the secondary structure of the RNA molecules from the PDB files. It also helps to visualise the 2D and 3D structures of the nucleic acids (DNA and RNA). The secondary RNA structure is predicted by the distance between atoms of a pair of nucleobases. X3DNA consider the relative position and orientation of the two base pairs (consider the position of $C3$ and $O2$ atom of each base) to generate secondary structures [131]. Here, we use X3DNA [88], one of the most accurate tools, to generate 2D RNA structures.

The similarities between the 2D RNA structures have been investigated by using base-pair comparisons. A more compact representation is needed to get more information from the 2D RNA structures. There is a correlation between 2D RNA representation and RNA function prediction. If we have more information on the 2D RNA structures and can encode these pieces of information in a well-structured form, we can determine their functions more accurately.

In this section, we introduce a new graph representation to encode 2D RNA structures. We, first, explore how to encode minimum free energy between bases as edge labels. Then we do experiments using this representation to find similarities between pairs of RNA strands to assess their functions. In this representation, we investigate whether MFE affects the prediction of RNA functions.

4.2.1 Sequence Free Energy

The minimum free energy (MFE) between base-pairs has been used for prediction of the 2D RNA structure by many works [7, 104, 127, 128]. A dynamic algorithm [126] was used to produce 2D RNA structure using MFE. The secondary structure is also derived from the nucleobase sequences using MFE [129]. Our approach is to use MFE to help determine the function of RNA.

In this section, we explore how we utilise MFE in a graph-structured form. In this representation, first, we intend to receive all free energy information embedded in the 2D RNA structural components, then we represent all these information in a graph-structured form.

The representation is approached in three categories. The first two are stem and loops, and the third one is the unpaired/straight part of the strand. We encode stem and loops using MFE on the components and we encode unpaired/straight part of the strand by representing base position as a regular tetrahedron. Our approach for 2D RNA graph representation is described as follows.

The minimum free energy between base pairs shown in Table 4.1 used to calculate the edge weight of the stem. Stem occurs in a paired part of the RNA strand. On the RNA structures, the most negative energy occurs between the most stable base pairs. All secondary RNA structures are predictive structures. We have used X3DNA [88] to construct base pair information, and we used Table 4.1, provided by the Vendeix et al. [132], to calculate edge weights of the stems. X3DNA is one of the classic and accurate tools to generate base pairs from the PDB files. The information from Table 4.1 is one of the most recent studies [132]

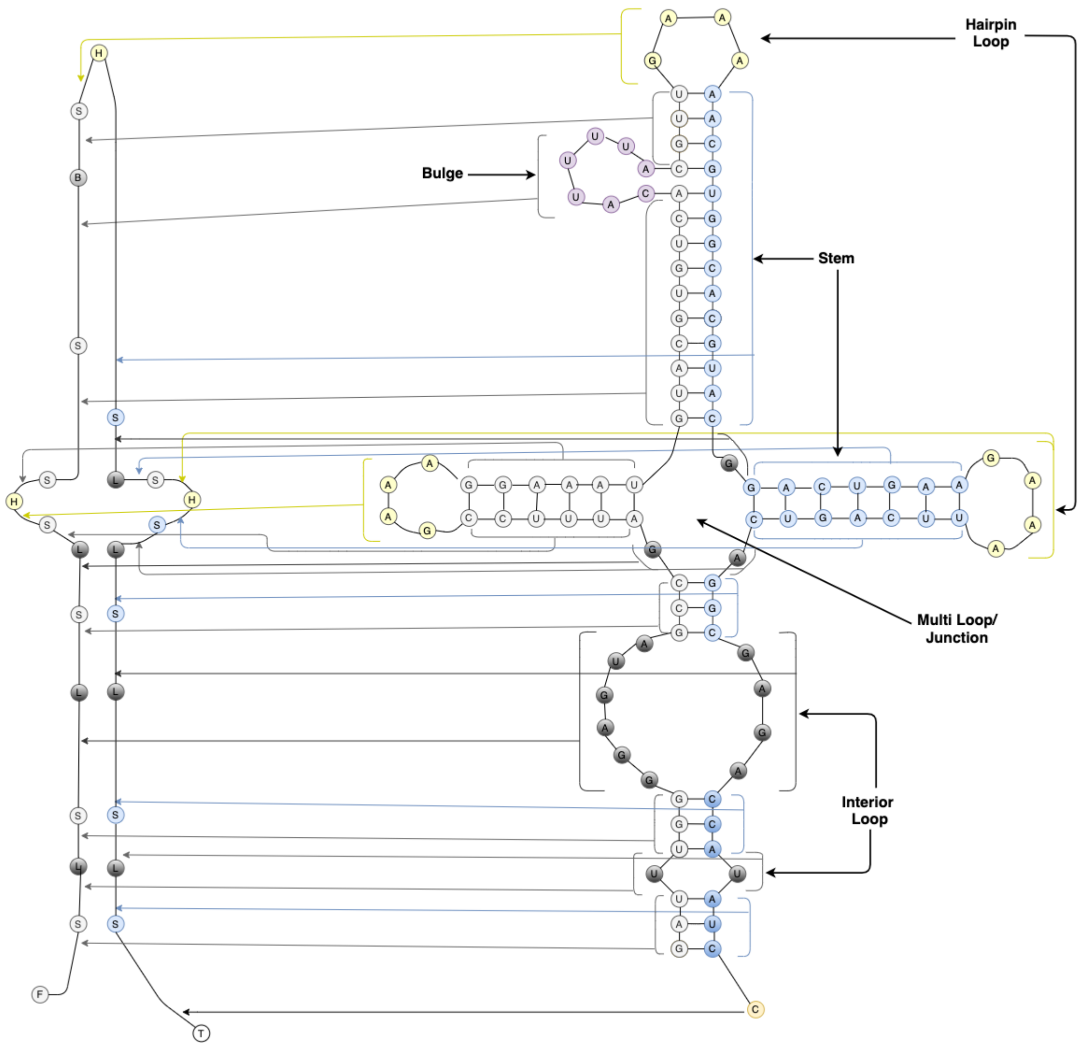


Fig. 4.1 5S RRNA secondary structure of the Escherichica coli Riboswitch derived from the 4Y1M.pdb. This proposed representation is using minimum free energy between base pairs as an edge weight. The left panel is generated from the right panel.

Base Pair	kcal/mol
C-G	-5.53
U-A	-4.42
U-G	-4.45
U-U	-5.82
U-C	-0.37

Table 4.1 Base pairs and minimum free energy between base pairs.

that provides MFE information on the paired bases. This study [132] only provides MFE on the stems.

The minimum free energy on the loops (hairpin, interior, and bulge) shown in the table Fig 4.2 is used to calculate the edge weight of loops. Fig 4.2 provided by Tinoco at al. [122]. To our best knowledge in the reviewed literature, this approach is one of the oldest technique, but it is still adopted nowadays for MFE computation [133]. As shown in Fig 4.2, there is no information for MFE on multi-loops (junctions). We treated multi-loops same as interior loops to compute the edge weights.

Nearly, 24 per cent of RNA chains in our data set [39] are straight chains. They have no secondary structure components, and they are unpaired. These shape of RNA chains significantly impact our ability to classify RNA molecules correctly. We used a different method to represent these straight non-paired RNA chains in the dataset, based purely on their base sequence. To calculate edge weights, we used a 3-base code, where each sequence of 3 bases is treated as a single edge (Fig 4.3), reducing the size of the representation by a factor of three. As represented in Fig 4.4, we encode each base like a regular tetrahedron as follows: A$(1,1,1)$, C $(1,-1,-1)$, U$(-1,1,-1)$, and G$(-1,-1,1)$. The distance between any base pairs are equal ($2\sqrt{2}$). Then we used the concatenation of a sequence of the position of the three bases. Because most graph kernels can only use a limited number of labels, we then reduced the set of 64 possible sequences to 4 by using k-medoids to build code-words of the code-book. And we applied Leaning Vector Quantisation (LVQ) to cluster edge weight in four classes from 1 to 4. As demonstrated on Fig 4.3, if we leave each edge weight like

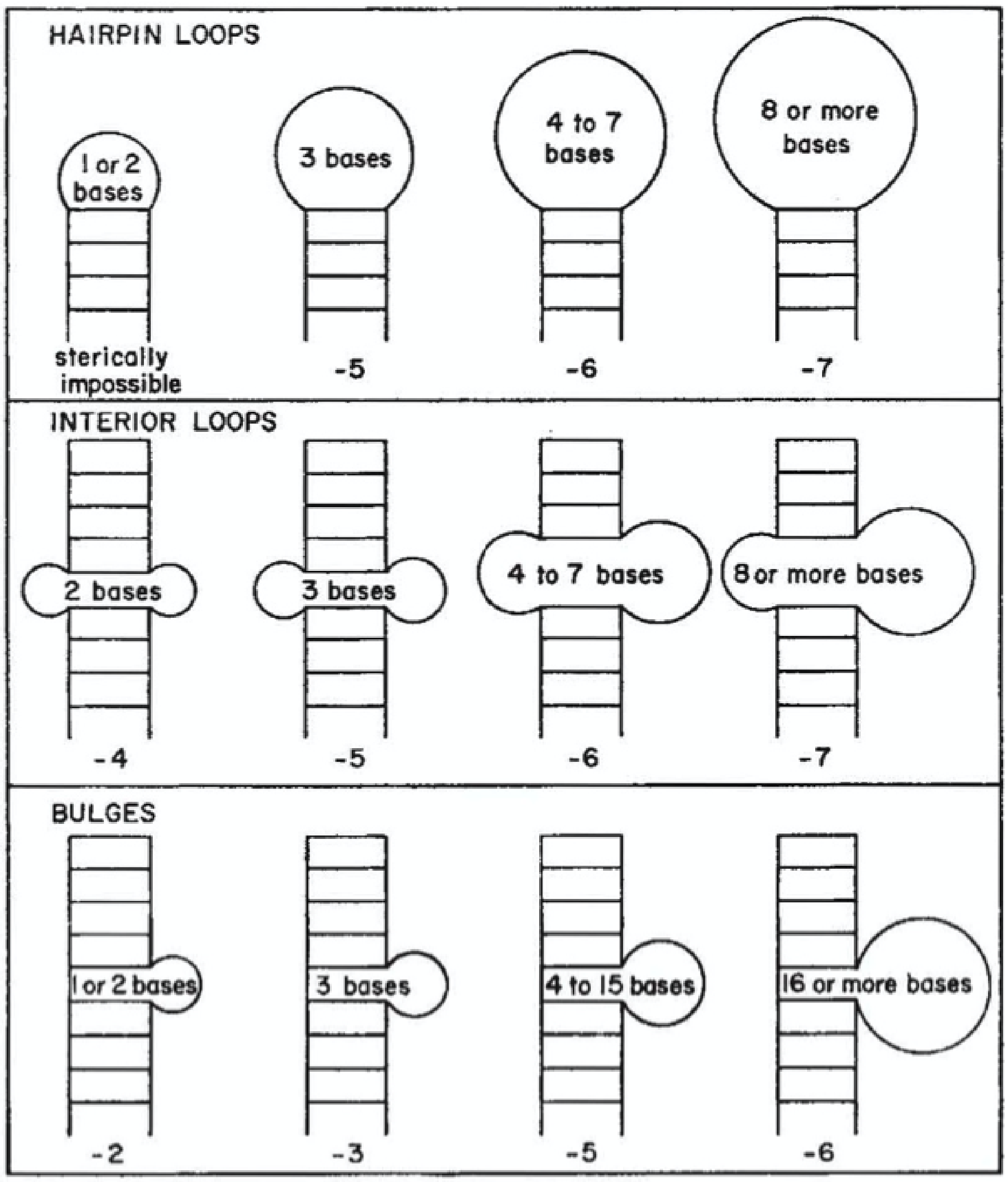


Fig. 4.2 This figure is reproduced from [122]. The information from the figure are used to calculate minimum free energy on loops.

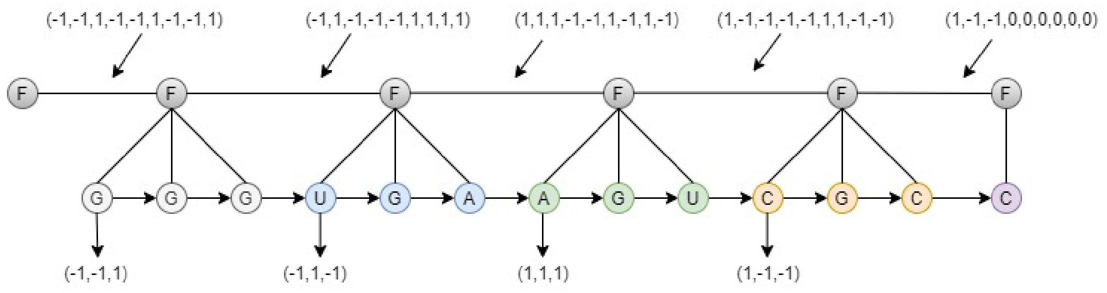


Fig. 4.3 Edge weight representation on the straight chains. Base sequence information extracted from the 1a3m.pdb chain B

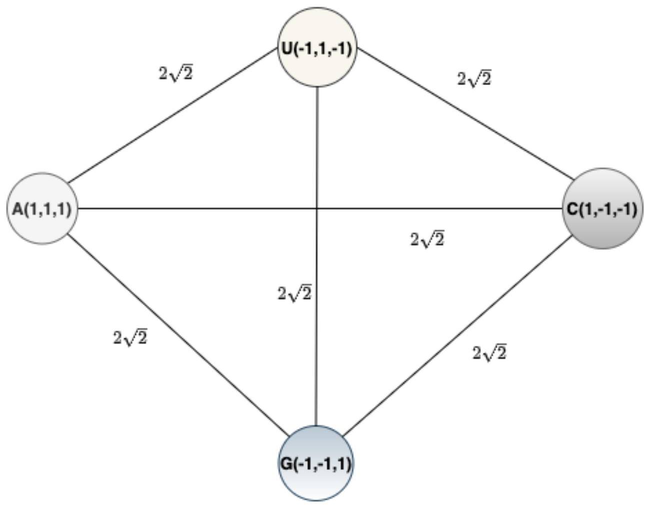


Fig. 4.4 Represent base position as a tetrahedron

that it would be 64 different edge weights, to minimise and to improve performance of the classification we encode each edge weight from 1 to 4.

Even paired RNA chains with secondary structure have some unpaired sequences (tails and queues) from 5’ to 3’ of the RNAs. These tails and queues also unpaired and straight part of the RNA chain. We applied the same method, mentioned in the previous paragraph, to tails and queues.

Eventually, in order to operate some graph kernels (for example WL kernels), we need discrete labels. We, therefore, did some refinement on the edge weights as follows. The edge labels from the 3-base code above directly map to the integer labels 1-4. For the structural edge weights from MFE, we checked if the absolute value of the edge weight is greater than

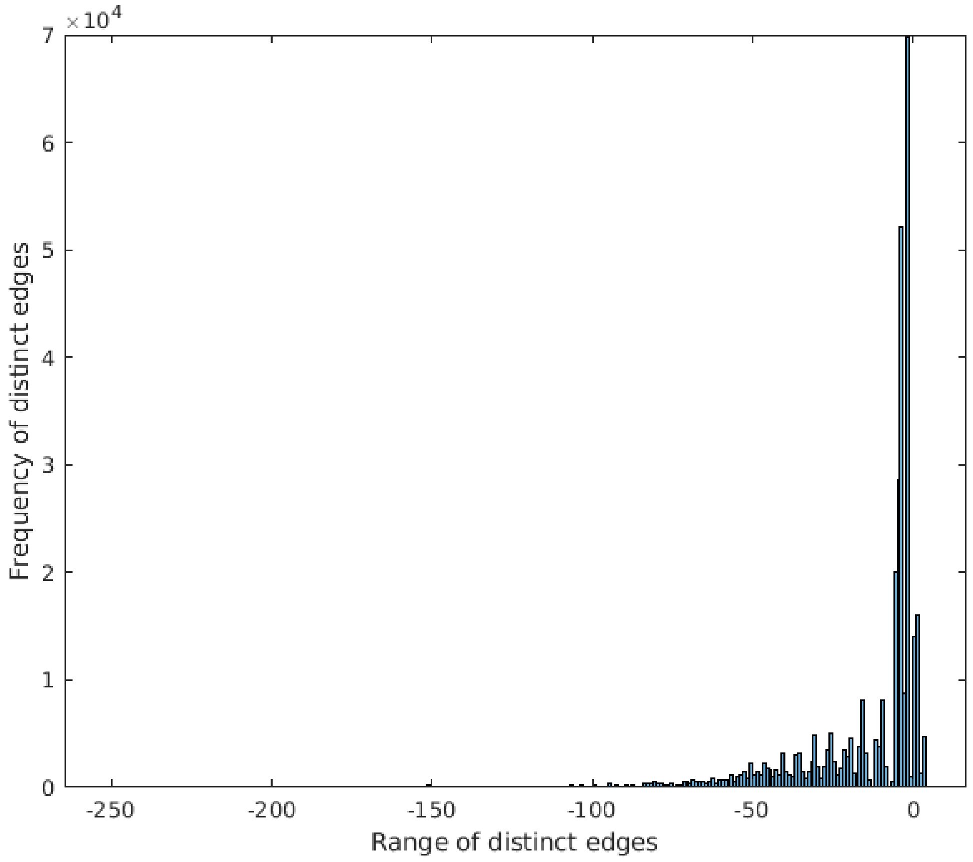


Fig. 4.5 Histogram of the edge labels before edge refinements. The distinct labels are in range of [-251.07,4.0]

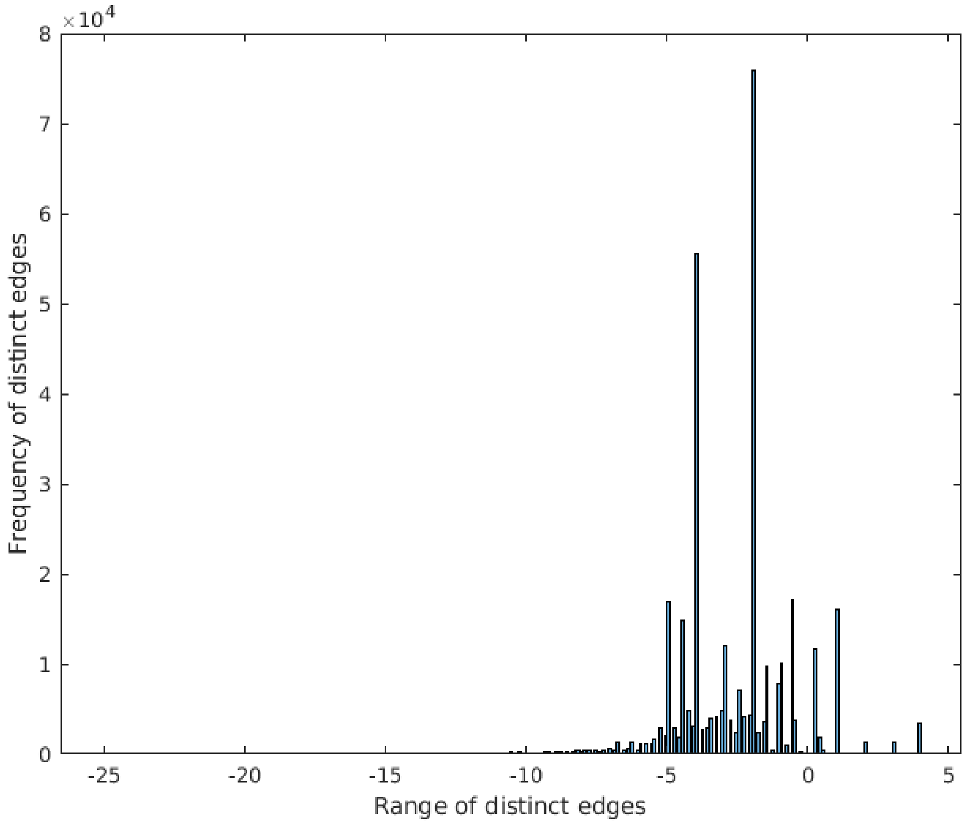


Fig. 4.6 Histogram of the edge labels after edges refinements. The distinct labels are in range of [-25.1,4.0]

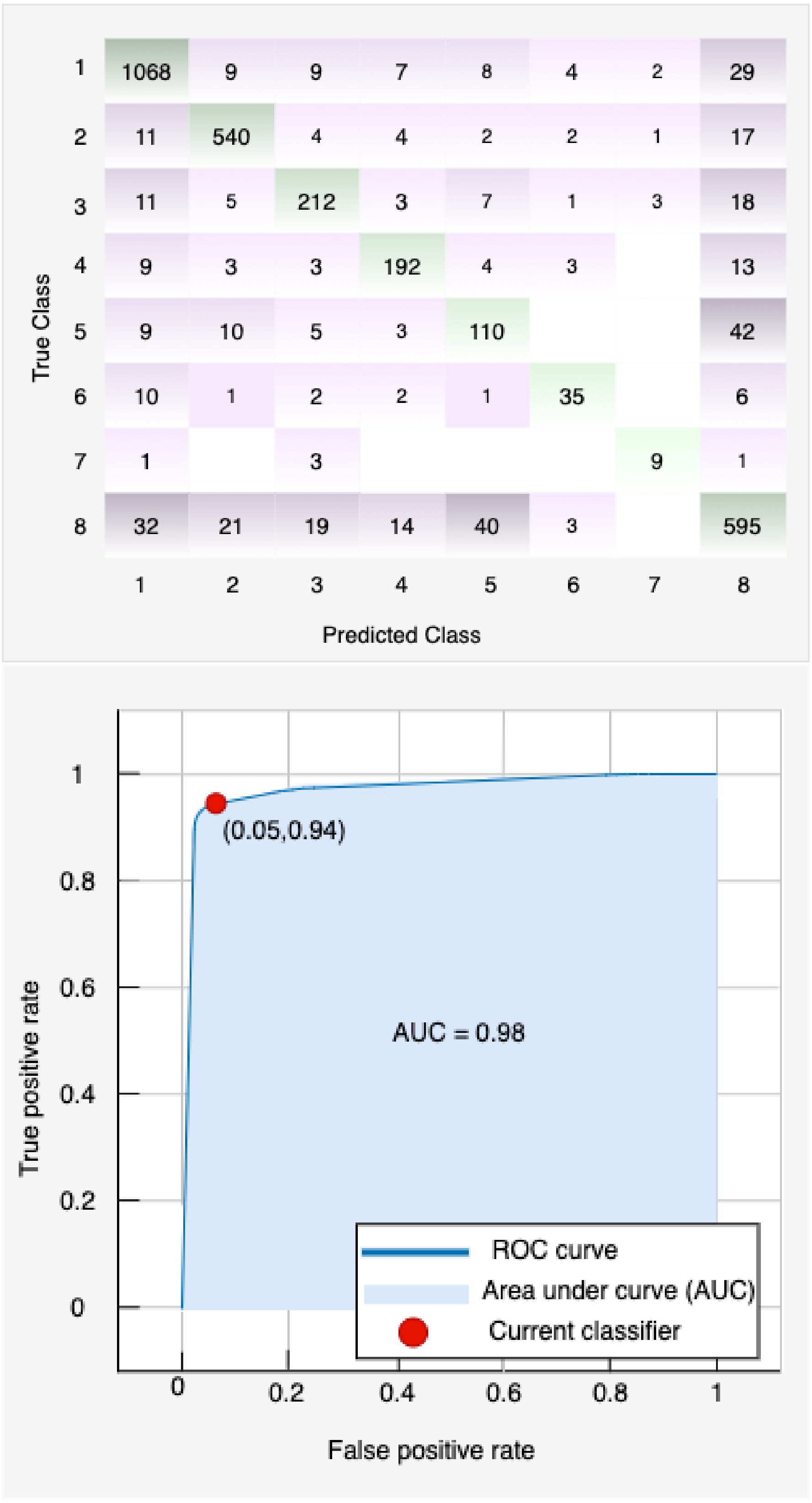
5, we divided it by 10. Then, we get all numbers that are rounded to tenths to leave only one number after the decimal points for encoding them as node labels. As represented in Fig. 4.5, the range of the discrete edge labels is too wide. This process intends to avoid too long numbers after decimal points, reduce the number of discrete edge labels, and make some normalisation on the edge weights. As presented in Fig. 4.6, after the refinement process, the range of the edge labels reduced to an appropriate range.

With this representation, we received relevant information from the chains, and we also decreased the size of the total vertices and edges from 1420903 to 348928 (approximately 75.4 per cent).

4.3 Result and Discussion

In this section, we provide the results obtained from the proposed 2D RNA graph representations. The state-of-the-art graph kernel methods utilised to predict the task of the RNA molecules from a given RNA graph classification data set.

Our goal for this analysis is to investigate whether the proposed representation is useful to classify RNA molecules efficiently, both in terms of reducing complexity and improving accuracy. We also explore the effect of different sources of information of the RNA include sequence, topology and our proposed representation (new representation NR) to classify RNA strands. The original graph representation of the RNA sequence consists of only base labels that the label of the nucleotide sequences are corresponding to their edge labels. The original graph representation of the topology of RNA includes the graph edges (including the sequence and base-pair edges) but no base labels. NR-W is our proposed weighted 2D RNA graph representation explained in section 4.2.1. NR-W includes MFE on each structural component of the RNA as an edge weight, and it also contains the concatenation of the position of the three base letter on straight-chain as an edge weight and each junction between adjacent structural component as a vertex. NR-UW is our introduced unweighted graph representation that only includes the label of each vertex on RNA chains. NR-UW does not include edge weights.



True Class \ Predicted Class	1	2	3	4	5	6	7	8
1	1068	9	9	7	8	4	2	29
2	11	540	4	4	2	2	1	17
3	11	5	212	3	7	1	3	18
4	9	3	3	192	4	3		13
5	9	10	5	3	110			42
6	10	1	2	2	1	35		6
7	1		3				9	1
8	32	21	19	14	40	3		595

Fig. 4.7 Confusion Matrix and ROC Curve was obtained by applying WL-OA method to the New Representation (NR-W)

	Seq only	Top only	Seq +Top	NR-W	NR-UW
WL-OA	92.0	73.1	**92.4**	86.9	77.5
SP	**91.3**	79.5	91.1	80.9	75.4
APC	**90.3**	85.4	89.9	79.5	79.4
SA	89.2				

Table 4.2 Classification accuracy for the RNA dataset using Weisfeiler Lehman Optimal Assignment (WL-OA), Shortest Path (SP), All-Paths and Cycle methods (APC), Sequence Alignment (SA), and a variety representations.

	Seq only		Top only		Seq + Top		NR-W		NR-UW	
	acc	speed	acc	speed	acc	speed	acc	speed	acc	speed
WL-OA	92.0	16m 32s	73.1	5m 03s	92.4	3h 18m 21s	86.9	2h 59m 44s	77.5	19s
SP	91.3	10h 36m 14s	79.5	10h 40m 10s	91.1	10h 43m 15s	80.9	13m 1s	75.4	10m 19s
APC	90.3	17m 57s	85.4	16m 33s	89.9	24m 20s	79.5	1m 46s	79.4	1m 9s

Table 4.3 Time taken to classify the RNA dataset using WL-OA, SP, APC, SA, and a variety representations. s: second, m:minute, h:hour

Then, we applied the graph-based method and machine learning classifiers to determine the accuracy of the most effective methods on each source of information. The machine learning classifiers applied; Bagged Trees, Cubic SVM, Quadratic SVM, and Subspace KNN. As shown in Table 4.3, in our experiments: Bagged Trees outperform best results with SP methods on NR-UW. Cubic SVM and Quadratic SVM outperform best results with WL-OA on NR-UW where PCA was on. Subspace KNN outperforms the results with APC methods on all representations; with SP methods on Seq, Top and NR-W; with WL-OA on NR-W. In the use of SP method, we shift edge weights from negative side to positive side by adding 26 to edge weights and get its integer value.

We also evaluated the classification performance by using the Receiver Operating Characteristic (ROC) curve. Using the Subspace KNN classifier, we observed the true positive rate (TPR) of 94 per cent. We also measured the area under the curve (RUC), Precision/Recall curve, which is around 98 per cent.

In our experimental results, we find that the reduced representation results in a decrease in performance relative to the full representations, in exchange for a considerable increase in speed. The best performance is obtained by using the WL-OA on the (Seq + Top) of the RNA,

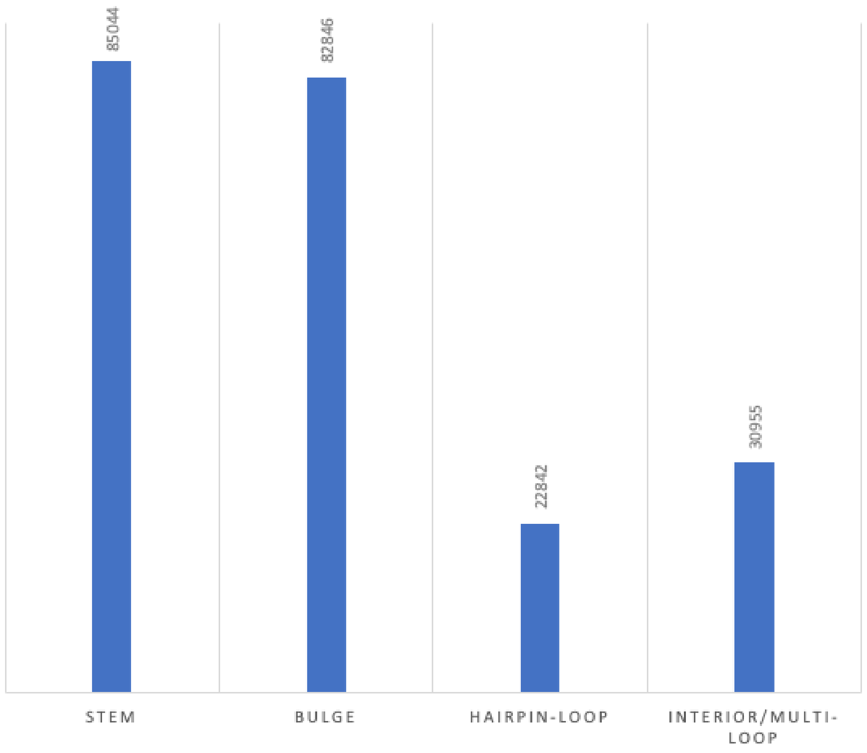


Fig. 4.8 Histogram of the 2D RNA structural components in RNA graph classification dataset

with an accuracy of 92.4 per cent; with our proposed representation (NR-W) the accuracy is 86.6 per cent, with a speed-up of. Our representation (NR-W) is also performing better than 2D RNA representation (topology only). The highest result obtained from 2D RNA representation was 85.4 per cent of the accuracy of rate using the APC method. The highest result obtained from our representation (NR-W) is 86.6 per cent using WL-OA method where our representation includes edge and sequence labels of the reduced graph while topology (2D RNA) only includes only edge information of the full graph. The test results from Table 4.2 and Table 4.3 proved that using MFE on the structural components as edge weights improve the classification performance. On the other hand, adding the base label to the topology (Seq + Top) performs better than our representation (NR-W).

When we remove edge weights from the graph representation, we still can classify RNA molecules with an accuracy of 79.4 per cent using the APC method. The results show that the weighted graph representations outperform the unweighted graph representations (NR-UW) in the application of three graph kernel methods (WL-OA, SP, and APC).

With proposed representation, we reduced the size of the graphs and increased the accuracy of the classification on the 2D structures. In this representation, our proposed method outperforms the existing 2D RNA representation (topology only). But still using primary RNA structure perform better than any secondary structure representations. We need a better RNA representation that aid to classify RNA chains in higher accuracy. In the next chapter, we investigate more sophisticated graph representations for encoding geometry shape of the RNA.

4.4 Summary

In this chapter, first, we state the representation problem. Second, we investigated the solution to address this problem. We looked at existing representations for encoding 2D RNA structure. Then we introduced a new RNA graph representation for encoding 2D RNA structure using a variety of methods. The method investigates whether MFE affects the prediction of RNA function. All RNA molecules are not in the same shape, and only encoding the structural components as a graph is not enough for prediction of the RNA functions. There might be different approaches to refine the representation for obtaining better performance on the RNA graph classification data set. Rarely, the RNA structures consist of some heads and tails. These infrequent part of the RNA structure is also needed to be encoded. And there is no free energy on these part of the chains. We brought alternative approaches to encode each of these uncommon parts of the RNA structure. Then we combined them to build a unit graph representation.

In our experiments, we successfully encoded MFE, embedded in the structural components of the 2D RNA, in a graph-structured form. We prove that using MFE as edge weights in a graph with some refinement on the infrequent part of the RNA structure is affirmative for RNA classification problem.

We applied the state-of-the-art graph kernel methods and a standard sequence-based method to measure the similarity between a pair of RNA molecules. The applied methods explained in Chapter 2. Then, we applied machine learning classifiers to classify these similarities into high-level classes. Finally, we compared the results obtained from our proposed representation to the baseline representations (the primary structure and secondary structure). As presented in Table 4.2 and Table 4.3, the best results from [39] use WL-OA on the RNA graph representation (primary structure) with an accuracy of 92.0%. With 75% fewer number vertices, we get the best result from the secondary RNA structure using WL-OA method (accuracy: 86.6%) on our representation. Our representation outperforms the best results from [39] use three graph kernel methods on the secondary RNA structure. The unweighted of our graph representation is also can be used to classify RNA molecules, with an increase in speed using WL-OA but a drop in accuracy.

The results obtained from the secondary structure both of our representation and existing representations do not perform better than the primary RNA structure. The results from the secondary RNA structures, both our proposed representation and 2D RNA representation from [39], do not outperform the primary RNA graph representation. This is because of several reasons. First, when we use 2D RNA representation from [39] we lose the information of the base label. For our representation, we also lose information when reducing the size of the graph. Second, for approximately 24 per cent of the data set does not include structural components, and that affects the classification of RNA molecules. Third, all secondary structures are predictive structures. The actual shape may differ. That also affects the performance of the classification. A better tool might be useful to generate secondary RNA structures to obtain better experimental results.

Chapter 5

Geometric Shape of the RNA (3D RNA)

In Chapter 4, we presented extensive research on 2D RNA structures for encoding them as graphs to determine the function of the RNA molecules. 2D RNA structures are largely dependent on sequence and Watson–Crick (WC) base pairs and non-WC base pairs. As explained in Chapter 4, using only extracted 2D RNA shape information is not sufficient to classify the whole dataset according to their functions. Approximately 24% of the RNA chains in our data set consists of a small number of nucleotides. These short and straight RNA chains have no base pairs and secondary structures. We added a sequence of the edges and node labels to the 2D topologies to encode all RNA chains in our RNA dataset to classify the entire dataset. In the previous chapter, we developed an extra encoding method for these chains, dependent on the sequence. However, it is likely that their 3D structure is important for their function.

A range of studies [11, 110, 134–139] use 2D RNA as an input to generate 3D RNA topologies. A serious drawback of these approaches is that short RNA chains have no structural motifs and secondary structures to produce 3D RNA topologies, and are neglected in the 3D RNA studies. On the other hand, it is difficult and computationally expensive to generate 3D RNA topologies. For instance, RAG-3D [134] can generate 3D models up to a maximum of 240 nucleotides length, and RNAComposer [11] can produce 3D RNA structures up to 500 nucleotides length. Whereas, the average nucleotide length of the RNA strands in our data set is 447.106, and the largest RNA chain in our data set consists of 4298

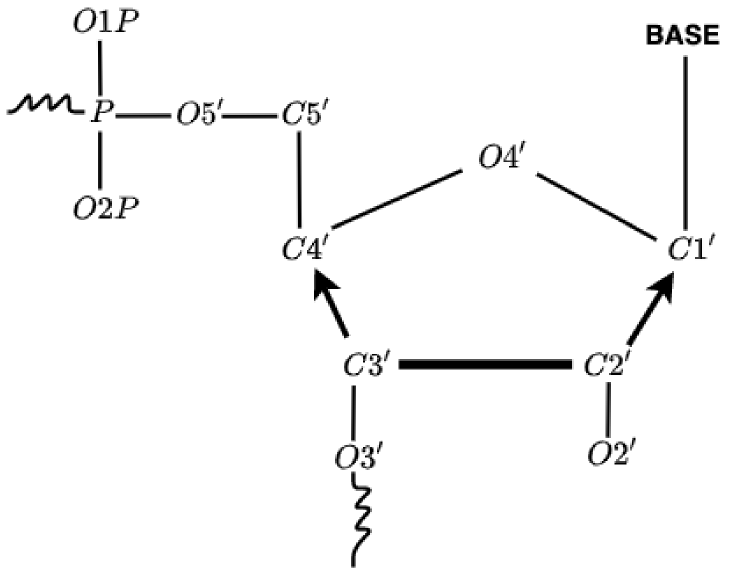


Fig. 5.1 A diagram of the nucleotide showing the $C3'$ atom [141].

nucleotides. Current methods are not sufficient to generate 3D RNA topologies of the large RNA chains. However, we need to encode 3D RNA structures as graphs that can apply to RNA chains of any size in a dataset.

In this chapter, we will investigate graph representation methods for encoding 3D RNA structures. These representations are based on geometric three coordinates (x,y,z) information of the $C3$ atom of the RNA backbone sugar. A positive aspect of using geometric information of the RNA shape is that it applies to all RNA chains of our dataset. It also assists to make a comparison between any pair of RNA molecules for classification purposes. To the best of our knowledge from the reviewed literature and our experiments, similar 3D RNA structures belong to the same functional RNA family [40, 39, 11, 14, 50, 140]. The function of the RNA molecules is critically related to their geometric shape of the backbone sugar.

For this reason, first, we will extract three coordinates (x,y,z) information of the $C3$ atom of each RNA nucleotide from PDB files. Second, we will use this coordinates information to represent the geometric shape of the RNA with geometric measures such as `'arc length'`, `'curvature'`, square root velocity function `'SRVF'`, and `'base position'`. Each RNA representation is a feature matrix, and each row of the matrix is a feature of $C3$ of an RNA nucleotide. Each matrix contains continuous feature channels. We will use k-medoids and Learning Vector Quantisation (LVQ) to cluster and label each row as a node. We will cluster node labels because graph kernel methods only accept a limited number of distinct node labels as input, and they can only be applied to 1D node features. For instance, the APC method only accepts 3 distinct node labels as input, and the node labels have to be 1D.

The purpose of these representations is to encode the information from the 3D RNA structure as a graph, then use these representations to address graph classification problems. The 3D shape of the RNA is the most complex structure. Using information from these highly structured molecules outperforms 2D RNA structures in RNA classification problems. Here, we present our third contribution to the research project, and the chapter is structured as follows. Section 5.1 explains the problems, our hypotheses, and our approaches. In Section 5.2, we investigate existing 3D RNA representations. In Section 5.3, we describe how to represent 3D RNA shape with `'arc length'`, `'curvature'`, `'SRVF'`, and `'base position'`. In section 5.4, we introduce our approaches for encoding the 3D RNA structure as a graph. In Section 5.5, we present and evaluate the performance of the graph kernel methods on our introduced RNA graph representations, where the results obtained from our RNA dataset. Finally, in Section 5.6, we summarise the chapter.

5.1 Problem Statement

In this section, we will define the representation problem for encoding the 3D shape of RNA as a graph for use in learning methods to determine their biological tasks in the cells.

In the reviewed literature, the geometric shape of RNA has been less studied [9] than the 1D/2D RNA structure to determine the functional type of the RNA molecules. Whereas, the $3-\textit{dimensional}$ (3D) shape of RNA is more complex than the secondary RNA structure and consists of rich structural information of RNA strands. The principal questions that arise are,

- How to encode these 3D RNA structures as a graph?
- How to make a pairwise comparison of these graphs, where the graph structures come in different shapes and sizes?

Our goal is to find powerful representations to encode 3D RNA structure as a graph, then use these representations to determine their functional types using learning methods.

In this chapter, we investigate four representation methods to transform 3D RNA structures into the graph-structured form. For comprehensive shape analysis, we treat each RNA strand as a 3D continuous curve. First, we represent each curve with square root velocity function (SRVF) to check the bending and stretching of the shapes. Using `'SRVF'`, preserves the principal characteristic of the curve during translation, rotation and scaling. Second, we represent 3D RNA shape with `'arc length'`. With this method, we find the length of the arc between two nucleotides along the RNA curve. Third, we apply `'curvature'` to the RNA curve to find how fast the direction of the curve changes with respect to `'arc length'` along the curve. Fourth, we use the `'base position'` method to find the minimum distance between a nucleotide and its nearest nucleotide. Finally, we apply k-medoids and LVQ to encode these representations as graph labels. We also generate joint representations using `'arc length'`, `'base position'` and `'curvature'`.

In these ways, these representations do not require getting information from the complex 3D RNA topologies. Instead, it receives the three coordinates (x,y,z) information of the $C3$ atom from each nucleobase of the RNA chains. These representations can be applied to RNA strands of any size. Consequently, we have two hypotheses as follows:

- **H1** In the use of learning algorithms for classifying RNA molecules according to their functional types, using information extracted from 3D RNA structures performs better than using information extracted from 2D RNA structures.
- **H2** In the use of learning applications, applying the `'arc length'`, `'curvature'`, `'SRVF'`, and `'base position'` to the (x,y,z) coordinate of each $C3$ atom of the RNA nucleobases and then representing each of them as a graph improves classification performance.

5.2 Related Work

To best of our knowledge from reviewed literature, 3D RNA structure less studied than 1D/2D RNA structures. There is two common methods for 3D RNA graph representations.

First, 3D RNA graph representation are investigated by using 2D RNA topologies and three coordinates (x,y,z) information of the RNA backbone sugar [134–136].These methods use a tool to generate 2D RNA topologies and extract three coordinate information from the PDB files to generate 3D RNA structures. This approach encode each structural component as node and edges between the structural components. They also add three coordinate information to arrange helices and add an additional node to the center of the junctions to produce 3D RNA graph representations.

Second, 3D RNA graph representations are investigated by clustering structural motifs of the 2D RNA [110, 137–139]. The motifs are the structural components of the 2D structure of RNA molecules and are presented in Fig. 2.2. The 2D RNA structural motifs are called hairpin, stem, bulge, interior loop, and multi-loop. This approach represent each group of the motifs as node and add edges between two groups if there is at least one matched motif exist.

5.2.1 Existing Representations

RNAJAG [136], RNA-Junction-As-Graph, represents junctions and helical regions of the RNA strand as a tree graph. RNAJAG uses an existing 2D RNA structure as an input and determines the junction topology in RNA, and then predict tree graphs of these RNA junctions. RNAJAG provides an ensemble of helical arrangements within the junctions to approximate 3D structures. The model constructs tree graphs using geometric information $(C1,C6,C8)$ of RNA backbone to position the edges in the junction area. The method is limited to large RNA structures.

RAGTOP [135], RNA-As-Graph-Topologies, represents tertiary RNA structures in a hierarchical graph structure to determine the 3D topology of riboswitches using given predicted 2D RNA structures. RAGTOP also utilizes RNAJAG [136] to predict junction topologies of the RNA strand to produce 3D graphs. The method preserves 2D topologies and adds geometrical information of the helical arrangements in 3D. RAGTOP represents 3D graphs using knowledge-based statistical potentials derived from the known RNAs bending and torsion measures of internal loops. The RAGTOP encodes loops (hairpin, internal loop, junction, and bulge) as nodes; helices in pseudoknot-free structures as edges, where edges also exist between adjacent loops and helix ends. RAGTOP assigns 3D coordinates for the vertices at the centres of helices and loops; and then adds pseudoknots interactions as additional edges.

RAG-3D [134], Rna As Graph 3D, is a web-based tool and a data set that provides the 3D structure of the RNA as a tree graph up to 10 vertices (≈240 nucleotides). Each graph consists of its subgraph building blocks. The web tool compares the graphs in the data set and their substructures to find similar topologies where the connectivity between vertices computed with Laplacian Matrix $(L = D - A)$. RAG-3D uses the label and number of vertices and the eigenvalue of the Laplacian matrix to find structurally similar graphs. The 3D RNA representations have connectivity with 2D RNA structures and 3D atomic coordinates (C1, C8 and C6) of the backbone sugar. RAG-3D uses RNAView [106] to produce the pseudoknots free secondary RNA structures to determine 3D building blocks. In this graph representation, helices represent edges, and other 3D points (loop regions) and 2D structural components represent vertices. RAG-3D also does not contain pseudoknots that mostly occur in large RNA chains. RAG-3D includes at least two vertices and does not encode straight chains.

RNA 3D Motif Atlas [137] uses VARNA [142] to generate 2D RNA structures, and use the structural components of 2D RNA structure to produce 3D motif groups. The motif groups are defined by structurally similar loops. RNA 3D Motif Atlas represents each 3D motif group as a node and a weighted edge if there is at least a motif in a motif group matched a motif in another motif group. Similar to the RNA 3D Atlas motif, VeRNAl [139], RNA Bricks [110] and CaRNAval [138] also represent 3D motifs into graph structured form to explore similarities between motif groups.

RNAComposer explained in [11], the 3D structure of the riboswitches are generated by using 2D RNA tree graphs, and for large RNA sequences, a significant challenge is required to predict 3D structures. RNAComposer is able to generate 3D RNA structures only up to 500 nucleotide length [11].

All these representations neglect straight RNA chains. For approximately 24% of RNA strands in our data set, they have no secondary structures. We need to explore 3D RNA graph representations that can apply RNA strands of any size. Elastic shape analysis [13, 9] treats each RNA strand as a three-dimensional curve and represents RNA with a square root velocity function (SRVF). This method is important for the geometry shape of RNA analysis. In Section 5.3.1, we will describe the `'SRVF'`, and in Section 5.4, we will describe how to encode the `'SRVF'` representations as graphs.

5.3 3D RNA Representation Methods

If we treat 3D RNA structures as 2D RNA structure, use their topologies, we lose lots of information from the RNA structures. Also it is not possible to compare small RNA chains which have no 3D and 2D structural components. In this section, we investigate 3D RNA representations using information taken from the geometry shape of RNA. We analyse four representation methods.

At first, we look at Elastic Shape Analysis to treat RNA as a 3D curve and represent RNA curve with square root velocity function (SRVF). Second, we analyse `'arc length'` to measure the distance between two specific points on the RNA strand and use these measurements to encode the 3D RNA structure as a graph. Here, each measurement is the `'arc length'` of two specific points on the RNA strand. Third, we investigate `'curvature'` to measure how fast the direction of the curve changes at specific points along the RNA strands. The specific points are the three coordinates (x, y, z) of the $C3$ atom of the backbone sugar of RNA molecules. Finally, we investigate the `'base position'` using the backbone position of the residue (the position of the $C3$ atom) and centroid position of the base.

5.3.1 Elastic Shape Analysis (ESA)

In this section, we introduce a novel graph representation for encoding the geometry shape of the RNA. We examine the geometric shape of the RNA structure using Elastic Shape Analysis (ESA). ESA treats 3D RNA structures as parameterized 3D continuous curves $\beta : [0,1] \to R^3$. The goal of ESA is to preserve the principal characteristic of the curve during translation, rotation, and scaling.

For a comprehensive shape analysis, we represent each 3D curve with a square root velocity function (SRVF) to find the bending and stretching part of the RNA shapes [37, 13]. We utilise the geometric backbone of the RNA chains in euclidean space (R^3) to parameterise the curve. To do this, we extract the sequence of three coordinates (x,y,z) information of $C3$ atoms of RNA strands from the PDB files. We represent the sequence of the $C3$ atom with its three coordinates as a curve ($\beta_i : [x_i, y_i, z_i] \in R^3$). Through the points $\{\beta_i\}_i^n$, we construct a continuous curve (β) under specific constraints that map the interval $[0,1]$ to R^3 and interpret the curve according to the (x,y,z) dimension of the $C3$ in euclidean space as follows.

$$\begin{aligned} t_1 &= 0 \\ for\ i &= 1 : n-1 \\ t_{i+1} &= t_i + \frac{1}{L}||\beta_{i+1} - \beta_i|| \end{aligned}$$

where n is the number of nucleobases in an RNA strand, L is the length of the RNA curve, and t_i is the time corresponding to point β_i, i.e. $\beta(t_i) = \beta_i$. We define $\beta(t)$ as a continuous curve $(\beta : [0,1] \to R^3)$. Now, we represent β with a SRVF $(q(t))$.

$$q(t) = \frac{\dot{\beta}(t)}{\sqrt{||\dot{\beta}(t)||}} \tag{5.1}$$

here, $||.||$ is euclidean norm, and $\dot{\beta}(t) = \frac{d\beta(t)}{d(t)}$ re-scales the length of the each curve β to 1. $\dot{\beta}(t)$ is computed from the known sample points (t_i, β_i) as

$$\dot{\beta}(i) = \frac{\beta_{i+1} - \beta_i}{t_{i+1} - t_i} \tag{5.2}$$

Time derivative also makes $q(t)$ invariant to translation of β [37]. $q(t)$ consist of both speed ($(q(t)^2 = ||\dot{\beta}(t)||$) and direction ($q(t)/||q(t)|| = \dot{\beta}(t)/||\dot{\beta}(t)||$) instantaneous of β in time t [13].

In conclusion, we represent each RNA chain as a curve with SRVF. Each curve inferred from the geometric shape of the RNA. In section 5.4, we will explain how we encode each of these SRVF representations into graph-structured form.

5.3.2 Arc Length

Arc length is the length between two specific points on a circle or curve. In this section, we introduce a novel graph representation for encoding the geometry shape of the RNA. We utilise `'arc length'` to encode RNA in the graph-structured form.

Here, s is `'arc length'` (the length of a segment of the circle), θ is a central angle (degree in radians), and R is a radius. As presented in Fig 5.2 on circle e, the `'arc length'` between P_1 and P_2 points is computed with the equation in radians, $s = R\theta$.

In our RNA application, we treat RNA strands as continuous curves. RNA curves are not perfectly round like a circle. In this case, we need alternative solutions to calculate the `'arc length'` between particular parts of the curve. For representing each curve as a graph, we split the curve into sub-intervals. In our application, each sub-interval is an arc between two nucleobases, where each RNA strand consists of a sequence of nucleobases, and each nucleobase consists of a $C3$ atom. We use three coordinates (x,y,z) information of the $C3$ atoms to compute the `'arc length'`s (the length of the sub-intervals) as following steps.

- *Step1:* We extract three coordinates (x,y,z) information of the $C3$ atoms from the PDB files. Treat RNA as a $3-\textit{dimensional}$ continuous curve that such curve $= (x_t, y_t, z_t)$;

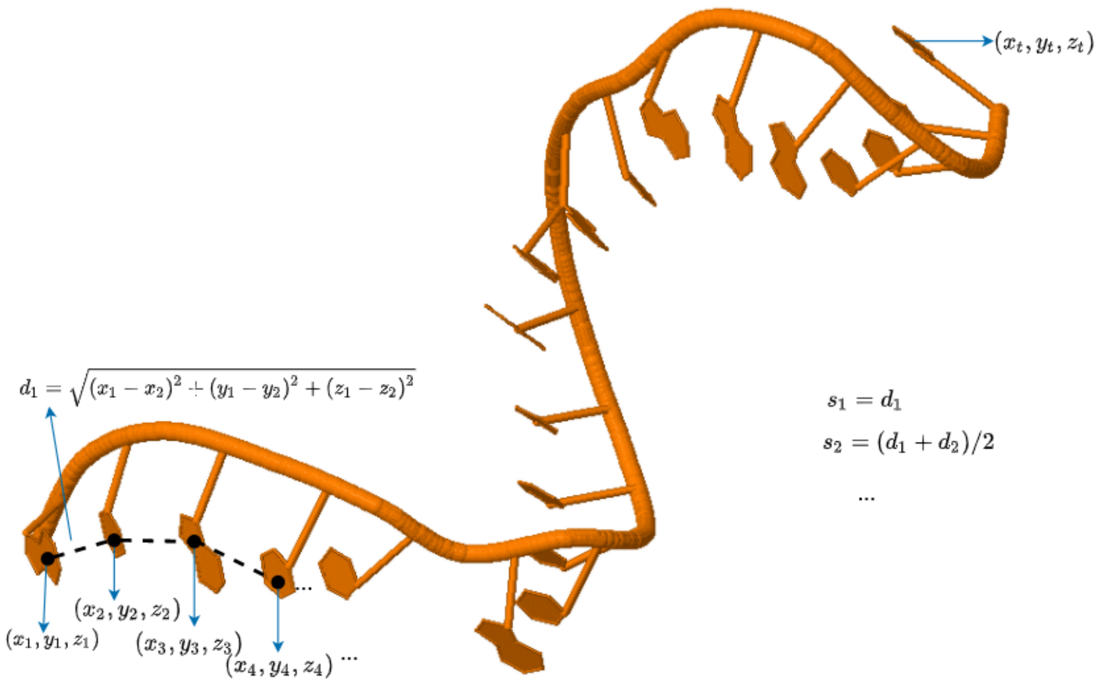


Fig. 5.2 3D RRNA backbone sugar trace derived from the 3BNR.pdb (chain D). The structure produced using MATLAB Molecule Viewer (Jmol)

- *Step 2:* We use Euclidean distance to measure the length of all sub-intervals.

$$for\ i = 2 : t$$

$$d_i = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2 + (z_i - z_{i-1})^2},$$

where t is the number of nucleotides in an RNA strand.

- *Step 3:* As presented in Fig. 5.2, We assign d_1 to s_1 ($s_1 \leftarrow d_1$)
- *Step 4:* We assign each midpoints to s that such

$$for\ i = 2 : t$$

$$s_i = (d_i + d_{i-1})/2$$

In this way, we find a series of `'arc lengths'` of specific points in RNA strands. In section 5.4, we will explain how we cluster these `'arc length'` values, and then how we encode them into the graph structured form.

5.3.3 Curvature

Curvature is a method of geometry that measures how quickly and sharply the direction of the curve changes [143–145]. Calculating `'curvature'` is more complicated than calculating `'arc length'`. In our RNA application, we want to measure how fast an RNA chain curved at the beginning to the end of every subinterval along the RNA curve. A `'curvature'` value relates to the part of how sharply the curve bends. A `'curvature'` value is higher in a more sharply changed part, and it is zero on the straight part of the curve. Here, we explain `'curvature'` with two methods.

The first method is to measure the `'curvature'` value using osculating circles. The `'curvature'` of the osculating circle is the `'curvature'` of a part of the curve, so the circle closest to this part approaches the curve best. As illustrated in Figure 5.3, the smaller circles are hugged by a sharper turned part of the curve, therefore the `'curvature'` value is greater in that tighter part; larger circles bend less sharply, and the `'curvature'` value is smaller. The straight part of the curve has a zero `'curvature'` value. The most snugly fit point of the curve is used to find the radius (R) of the circle. The circle centred on the inner side of the curve (concave). The magnitude of the curvature is equal to divide 1 by the radius of the circle ($\kappa = \frac{1}{R}$). The lower radius of the circle (R) means the higher `'curvature'` and the higher radius of the circle means the lower `'curvature'`.

A second method is to compute `'curvature'` using a parametric function `'arc length'` ($\vec{s}$) [146]. As presented in Fig 5.3, the direction of the tangent vector is not related to its magnitude, which is only related if the curve turns/bends. This method uses a unit tangent vector that rotates along the curve. The direction of the curve at point P_1 is presented with direction of the unit tangent vector and can be measured by using the inclination angle (ϕ). From point P_1 to P_2, the changed direction is related to `'arc length'`. Using the derivative of the unit tangent vector associated with small changes in `'arc length'` tells us

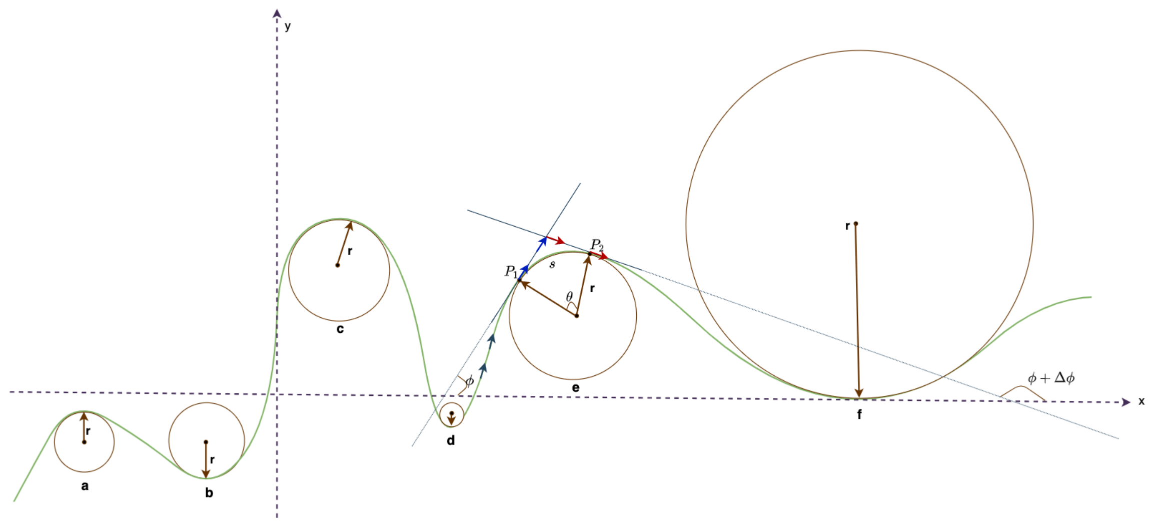


Fig. 5.3 Smaller circles are hugged by a more sharply turned part of the curve, and `'curvature'` is higher in that part ($\kappa = \frac{1}{R}$) . Larger circles are hugged by a less sharply turned part of the curve. Therefore, `'curvature'` (κ) is higher at d, and smaller at f part of curve in above figure.

the changes in the curve over time. The unit tangent vector controls how fast the direction of the curve changes from a given point to a second point along the curve. As seen in Fig. 5.3, blue and red unit tangent vectors show how fast the direction of the curve changes from P_1 to P_2, which is the rate of change of $\Delta\phi$ with respect to `'arc length'`. The derivative of the unit tangent vector to `'arc length'` is defined to be the `'curvature'`. The equation is as below:

$$\kappa = ||\frac{dT}{d\vec{s}}|| \tag{5.3}$$

where T is a unit tangent vector, $\vec{s}$ is `'arc length'`, and κ is `'curvature'`. To proof the Equation 5.3, we divide the derivative of unit tangent (T) with respect to the time 't' to the derivative of `'arc length'` with respect to 't'. Then, we measure `'curvature'` as below:

$$\kappa = ||\frac{dT}{ds}|| = \frac{||\frac{dT}{dt}||}{||\frac{d\vec{s}}{dt}||} \tag{5.4}$$

In our RNA application, we treat each RNA chain as a curve, and then we measure the `'curvature'` of each tiny `'arc length'` of the sub-interval of the curve. We used two functions explained in [147] to measure the magnitude of the curvature. The first function is the circumcenter function, which is used to find the radius R of the circumscribed circle of a triangle in 3D. The second function is the curvature function, which uses the circumcenter function to produce the radius of each triplet $[P_{i-1}, P_i, P_{i+1}]$ along the curve to measure the curvature vector at each point P_i. The curvature vector at $P_i = [x_i, y_i, z_i]$ is defined as $\kappa_i = T_i/R$. In Section 5.4, we will explain how we cluster these `'curvature'` values and how we encode each of them in a graph-structured form.

5.3.4 Base Position

In [39, 40], we introduced graph representation for encoding the geometry shape of the RNA. We used the information of tertiary RNA structure inferred from the PDB files. We considered two bases to be paired if they are compatible (A-U or C-G) and their end-points are within 4Å. The vertices were labelled with a geometric `'type'` label; 1 if it is part of a base pair, 2 if it is unpaired but within 6.5Å of another base, and 3 otherwise. Labels 2 and 3 are designed to distinguish bases on the inside and outside of loops. Each part was denoted by an edge in the graph between the adjacent vertices and paired vertices. Two sets of 3D coordinates were also provided, firstly for the backbone position of the residue (the position of $C3$ atom') and secondly for the centroid position of the base.

We also investigate graph representation for encoding the geometric shape of the RNA using the backbone position of the residue (the position of the $C3$ atom) and centroid position of the base. We, first, extract the distance between any pair bases in an RNA strand. Then, record the minimum distances between each atom and its nearest neighbour. To label these min distance values as vertex labels, we use k-medoids and Learning Vector Quantisation to cluster them. Thus, we label vertices with a geometry `'base position'`.

5.4 Encoding RNA Representations as Graph

In Section 5.2, we investigated how to represent the geometry shape of the RNA with `'arc length'`, `'curvature'`, `'SRVF'`, and `'base position'`. These representations consist of continuous feature channels. In this section, we encode these representations into the graph-structured form. We need $1-\textit{dimensional}$ distinct node labels to use in graph kernel methods for classification problems. We use learning vector quantisation (LVQ) to cluster node labels.

K-means algorithm minimizes the distances between data points to find the center of each cluster. The k-means algorithm first divides the input data into k groups, then calculates the centroid of each group using the average of all data points in each cluster, and then assigns each point to the one closest to the centre. In the use of k-means, the centroid of the cluster is not one of the actual points, but the mean of the points in each cluster. K-medoids [148] is a clustering algorithm similar to the k-means. Unlike k-means, the k-medoids algorithm chooses an actual data point closest to the center of the cluster as the center of the cluster.

Learning Vector Quantization (LVQ) algorithm is a supervised data clustering method that contains a set of codebook vectors [149, 150]. LVQ uses a set of given codebook vectors to determine each data point that is closest to them. The codebook vectors are appropriate measures of distance [150]. In the use of LVQ, the most important thing is to find efficient measures of distance. We used k-medoids to find the most appropriate measures of distance as follows.

At first, we separately apply `'arc length'`, `'curvature'`, `'SRVF'`, and `'base position'` to three coordinate (x,y,z) position of the $C3$ atom of each nucleotide in each RNA strand. Second, we build $(n\times m)$ dimensional matrix where n is the number of nucleotides of an RNA strand, and m is the number of feature channels (n: number of rows; m:number of columns). Third, we merge all produced $(n\times m)$ matrix and generate a new $(1420903\times m)$ matrix. Here, 1420903 is the number of entire nodes in our RNA data set. We apply k-medoids to each $(1420903\times m)$ matrix to find k suitable measures of distance as input codebook vectors. Finally, we use LVQ to find the most appropriate distinct node labels. Therefore, we use k-medoids and LVQ to cluster each row of the $(n\times m)$ matrix for encoding them as node labels.

Thus, each graph representation consists of $1-\textit{dimensional}$ k distinct node labels. We also represent edges between adjacent nodes and paired nodes. We used X3DNA [88] to generate node pairs. In these representations, (A, B, C) refers to `'arc length'` (A), `'base position'` (B), and `'curvature'` (C) respectively. In the following, for each representation, the most appropriate k value is chosen after many trials.

- RNA_ABC: In this representation, first, we separately represent each RNA strand with `'arc length'`, `'base position'`, and `'curvature'` to build a joint RNA representation. Second, we build an $(n \times 3)$ matrix for each joint representation, where the first column is `'arc length'`, the second column is `'base position'`, and the third column is `'curvature'`. Then, as explained above, we applied k-medoids using the k value as 6, and LVQ respectively to encode each row of matrix as a node label. Finally, we encode each RNA strand as a graph. This graph representation consists of 6 distinct node labels.

- RNA_AB: In this representation, we have a similar approach to RNA_ABC. First, we separately represent each RNA strand with `'arc length'` and `'base position'` to build a joint RNA representation. Second, we build $(n \times 2)$ matrix where the first column is `'arc length'`, and the second column is `'base position'`. Then, we applied k-medoids using the k value as 5, and we applied LVQ as explained above. Finally, we encode each RNA strand as a graph. This graph representation consists of 5 distinct node labels.

- RNA_AC: In this representation, we have a similar approach to RNA_AB. First, we separately represent each RNA strand with `'arc length'` and `'curvature'` to build a joint RNA representation. Second, we build $(n \times 2)$ matrix where the first column is `'arc length'`, and the second column is `'curvature'`. Then, we applied k-medoids and LVQ respectively. This graph representation also consists of 5 distinct node labels.

- RNA_A: In this representation, we had a similar approach. We represent each RNA strand with `'arc length'`. Then we encode node label information received by using `'arc length'` from the base position and cluster them using k-medoids and LVQ. This graph representation consists of $k = 5$ distinct node labels.

- RNA_C: This representation is similar to RNA_A. Here, we use `'curvature'` representation to encode RNA as a graph that consists of a maximum of $k = 5$ distinct node labels.

- RNA_B: This representation is similar to RNA_A. Here, we use `'base position'` representation to encode RNA as a graph that consists of a maximum of $k = 6$ distinct node labels.

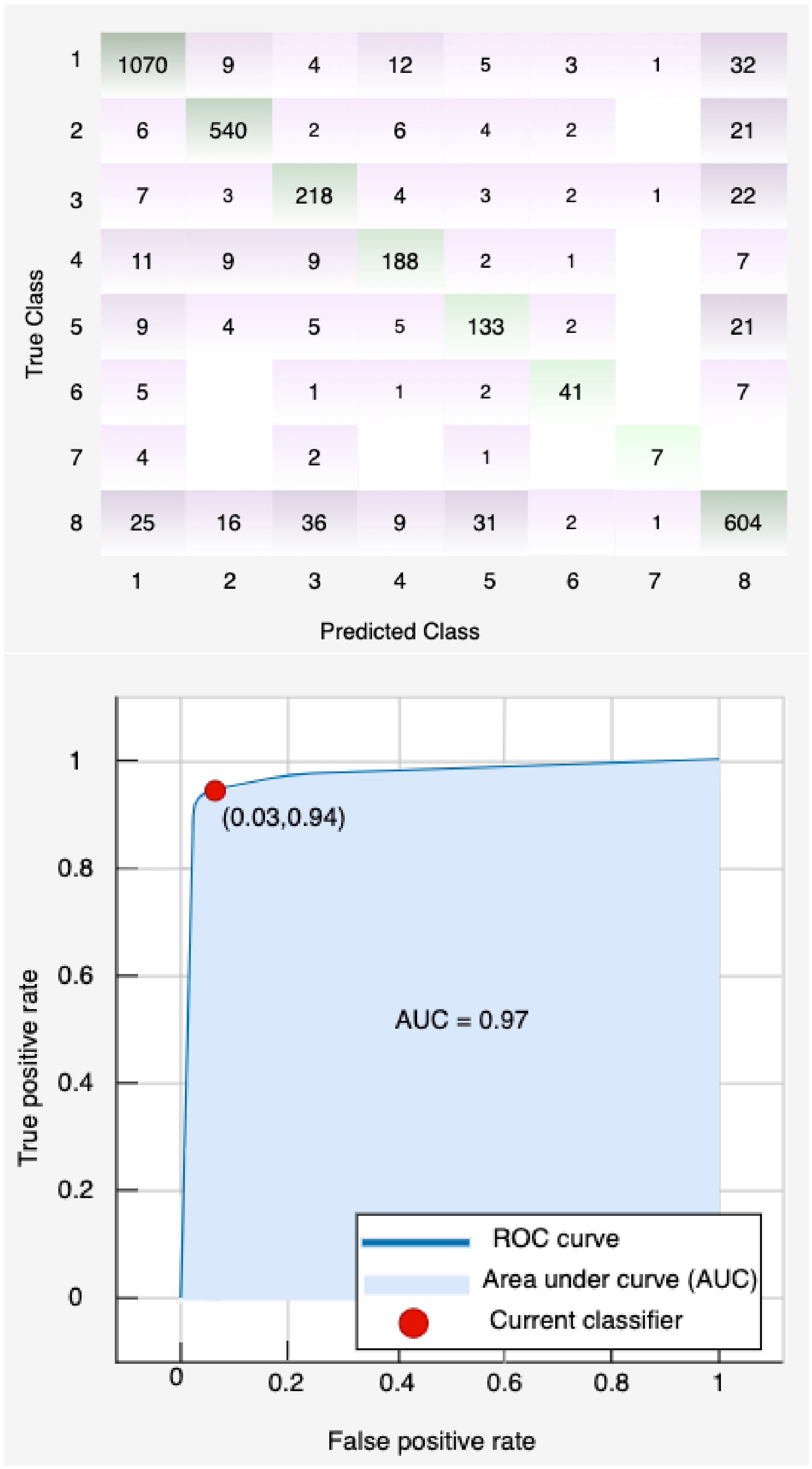
- RNA_SRVF: In this representation, first, we represent each RNA strand with `'SRVF'`, and then apply k-medoids and LVQ to encode `'SRVF'` representation as a graph. This graph representation consists of $k = 4$ distinct node labels.

- RNA_SRVF_P: Here, P refers to the principal component analysis (PCA). In this representation, first, we applied PCA to three coordinate information of the $C3$ atom of RNA strands, and then we used the same method as RNA_SRVF to encode the geometry shape of the RNA as a graph. This graph representation also consists of $k = 4$ distinct node labels.

5.5 Result and Discussion

In this section, we analyse the discriminative power of introduced 3D RNA graph representations in the application of Graph Kernels for RNA classification. The state-of-the-art graph kernel methods used to obtain the result of the classification accuracy.

Our goal is to evaluate which 3D RNA graph representation methods are most efficient to classify RNA molecules. In Chapters 3 and 4, we introduced 1D and 2D RNA graph representations. To the best of our knowledge from the literature, there are no standard or original graph representations for encoding only the geometric information of the RNA. In this section, we compare the 3D RNA graph representation methods introduced in Sections 5.3 and 5.4. We also make joint representations. We add base-pair connectivity as edges, 2D RNA graph topology, into the 3D RNA graph representations to evaluate the classification performance.



True Class \ Predicted Class	1	2	3	4	5	6	7	8
1	1070	9	4	12	5	3	1	32
2	6	540	2	6	4	2		21
3	7	3	218	4	3	2	1	22
4	11	9	9	188	2	1		7
5	9	4	5	5	133	2		21
6	5		1	1	2	41		7
7	4		2		1		7	
8	25	16	36	9	31	2	1	604

Fig. 5.4 Confusion Matrix and ROC Curve was obtained by applying SP method to the RNA_ABC

	R_A	R_C	R_B	R_AB	R_AC	R_BC	R_ABC	R_SRVF	R_SRVF-P	R_type
WL-OA	82.3	85.0	87.0	87.4	79.0	87.5	**87.6**	81.2	81.7	86.8
APC	76.1	82.6	84.7	85.6	76.1	85.7	**86.1**	80.1	81.4	84.3
SP	83.4	85.1	87.2	87.8	81.0	87.6	**88.1**	80.9	82.6	86.7

Table 5.1 To classify the RNA dataset using graph kernel methods (WL-OA, SP, APC) and our introduced variety of 3D RNA graph representations. All representations above include $1-dimensional$ distinct node features.The representations above: R_BC, R_SRVF and R_SRVF-P consist of 4 distinct node labels; R_A, R_AB, R_C and R_AC consist of 5 distinct node labels; R_B, and R_ABC consist of 6 distinct node labels. R_type described in Chapter 3 (Section 3.3).

+ 2D topology	R_A	R_C	R_B	R_AB	R_AC	R_BC	R_ABC	R_SRVF	R_SRVF-P	R_type
WL-OA	78.7	85.7	86.9	**87.4**	82.4	**87.4**	**87.4**	81.6	81.6	87.1
APC	76.1	82.9	84.6	85.3	76.1	85.9	**86.2**	80.1	81.5	85.5
SP	81.9	84.9	87.8	**88.1**	82.7	87.7	87.5	82.9	83.3	86.7

Table 5.2 To classify the RNA dataset using graph kernel methods (WL-OA, SP, APC) and our proposed variety of representations. The representations above include $1-dimensional$ distinct node features. R refers to RNA.

Then, we applied the graph-based method and a machine learning classifier (Subspace KNN) to determine the accuracy of the most effective methods on each 3D RNA graph representation. We also evaluated the classification performance by using the ROC curve. Using the Subspace KNN classifier, we observed the true positive rate (TPR) of 94 per cent. We also measured the area under the curve (RUC), Precision/Recall curve, which is around 97 per cent.

As explained in section 5.4, single representations contain only one feature information of RNAs. For instance, RNA_A is a single representation including only `'arc length'` features. Joint representations contain multiple features of the RNA strands. For instance, RNA_AC is a joint representation that includes both `'arc length'` and `'curvature'` features of the RNA strands. In our observations, we find using only joint representation methods are more powerful than using single representations. As presented in Table 5.1, the best performance is obtained by using the SP method on the R_ABC, with an accuracy of 88.1 per cent. SP method has a superiority to other methods on 8 out of 10 representations.

As the result presented in Table 5.2, in our experiments, adding 2D topology to 3D representations helpful on R_B, R_AB, R_AC, R_BC, R_SRVF, R_SRVF-P in the use of SP method (perform better on 6 out of 10 representations). The highest result obtained from R_AB representation 88.1 per cent of the accuracy of rate using the SP method. Adding 2D topology to 3D representations helpful on R_C, R_AC, R_BC, R_SRVF, R_type in the use of WL-OA method (perform better on 5 out of 10 representations).

We made a comprehensive shape analysis of the 3D RNA structure and applied the state-of-the-art graph kernel methods to decide which representation method is more efficient to classify RNA molecules. The graph kernel methods explained in Chapter 2 (see Section 2.3). In Chapter 6, we investigate Geometric Deep Learning (GDL) methods to determine the biological function of the RNA using 3D RNA graph representations introduced in this Chapter, and 1D RNA graph representation explained in the previous Chapters. We will also make a comparison between GDL methods and Graph Kernel methods for RNA classification for problems.

5.6 Summary

In this chapter, first, we state the representation problem. Second, we investigated existing representations for encoding 3D RNA structure. Then we introduced new 3D RNA graph representations for encoding 3D RNA structure using a variety of methods. We represented the geometry shape of RNA with `'arc length'`, `'curvature'`, `'SRVF'`, `'base position'` and encode these representations into graph-structured form. Finally, we presented our results obtained from our introduced 3D RNA graph representations using graph kernel methods.

In our experiments, we successfully encoded 3D structure as a graph with a variety of representations. We test our 3D RNA graph representation with state-of-the-art graph kernel methods. Our experiments prove that using introduced 3D RNA graph representations are help to determine the function of the RNA molecules with a high rate of accuracy.

In our extensive shape analysis in the application of graph kernel methods, using 3D RNA graph representations outperform 2D RNA graph representations. But still, 1D RNA graph representation has superiority on both 2D and 3D RNA graph representations (see Table 3.5, Table 4.2, and Table 5.1).

Chapter 6

Geometric Deep Learning

In Chapters 4 and 5, we proposed representation methods for encoding 2D and 3D RNA structure as a graph. In these chapters, we introduced various RNA graph representations. To use complex data in graph kernel applications for learning, we need powerful graph representations. Using kernel methods only allow us to apply them on a limited number of distinct node labels, and the node labels have to be 1-dimensional node features. For instance, the Shortest Path method [62] is not allowed to use negative edge labels, the All Path Cycle [70] method is only allowed to use a maximum of three distinct node labels.

In this Chapter, we analyze a variety of RNA graph representation methods. The graph representations consist of $(1-2-3)-dimensional$ continuous node features. We also introduce RNA 3D point cloud representations. The 3D Point cloud representations include three coordinates (x,y,z) information of the $C3$ atom and three additional channels. The additional channels are our proposed three RNA graph representations node features explained in the previous chapters. The nodes have $1-dimensional$ distinct features. Then, we apply state-of-the-art geometric deep learning methods to our RNA representations for classification purposes. We use geometric deep learning methods from two different approaches. First, we investigate graph neural networks to apply them to our introduced graph representations. We also apply graph neural networks to our graph representations introduced in the previous chapters. Second, we investigate deep learning methods on 3D point clouds to apply them to our introduced RNA 3D point cloud representations.

Traditional CNN models are allowed euclidean data structures as in input. The convolutional operations are applied on 2D or 3D grids. For instance, in image data, the number of the neighbourhood of each pixel is the same. They have regular grid structures. The CNN layers are able to extract high-level features from the hierarchical design of the grid data. In the convolutional layer, the same trainable localized filters can scan multiple locations of the regular data.

On the other hand, in the real world, all data are not in the same dimensions. For instance, the graph-structured data consist of a different number of vertices, and even each vertex in a graph is also has a different number of neighbours. It is not straightforward to directly apply CNN methods to the non-Euclidean graph data. The same trainable localized filters cannot be directly applied to the multiple locations of non-regular graph data. We investigate the modern graph convolutional networks (GCNs) that can be applied to non-regular graph data. The motivation of GCNs is to aggregate and transfer information from the node neighbours to update the nodes in a graph and to learn representations for network embedding.

The chapter structured as follows. Section 6.1 explains the problems and our approaches. Section 6.2 explains five graph neural network methods and the works related to graph neural networks. We explain how we applied these methods to our York RNA Data set. DGCNN [22] describe convolutional operation on the spatial neighbours. Graph Isomorphism Network (GIN) [25] first transform input data into vector representations, and then design machine learning methods for these vector representations. Structure2Vec [24] propose a new function to aggregate information from the neighbours and transfer them into the network embedding. gUNets [28] proposes two pooling operation (gPool, gUnpool) for up-sampling and down-sampling. LCGNN$_{GIN}$ [29] has a contribution to the GIN method. In Section 6.3, we explain PointNet [33], PointNet++ [34], PointConv [35] methods, and the works related to deep learning methods on 3D Point clouds. In Section 6.4, we explain our proposed representation. In Section 6.5, we show a framework that presents the data set preparation and encoding methods. In Section 6.6, we present the result obtained from the learning methods in our RNA representations and evaluate the results and methods. Finally, in Section 6.7, we summarise the chapter.

6.1 Problem Statement

In this section, we first define RNA graph representation problems where nodes include $(1-2-3)-\textit{dimensional}$ continuous features. Second, we will define the RNA 3D point cloud representation problems. Then we will explain the problems of using these representations in Geometric Deep Learning for RNA classification problems.

In Chapter 5, we proposed graph representations to encode 3D RNA structure as a graph. Then, we applied graph kernel methods to determine the function of RNA molecules. Our proposed RNA representations surpassed the baseline 3D RNA graph representation (RNA_type) in the application of the graph kernel methods.

Some graph kernel methods transfer the input graph to a new graph (a new graph can be a shortest path graph, WL graph, etc.),then uses a base kernel to make comparisons. For example, the Weisfeiler Lehman (WL) method iteratively updates the node labels according to the neighbour's nodes labels. The first goal is to refine node labels and then use a base kernel to make a pairwise comparison to check whether isomorphism between the graph pairs. Other graph kernel methods directly uses the original graph as input (APC method does not transfer the input graph to a new graph).

The first negative aspect of the graph kernel method is that the kernel method produces an $n \times n$ matrix where n is the number of the graphs in the data set. For large data sets it requires too much memory for computation. The second negative aspect is that the kernel methods not allowed to use multi-dimensional and continuous node features as input. A principal question that arises,

- How to encode the 3D RNA structure as a graph where nodes consist of $\textit{multi}-\textit{dimensional}$ continuous features?

Graph neural network methods are inspired by the graph kernel methods, specifically by the WL kernel method. The modern graph neural network methods aggregate messages from the neighbour nodes and learn representations for network embedding. The power of graph neural network methods is related to how the knowledge from neighbouring nodes gathered effectively for node embeddings.

In this chapter, we investigate the problem of representation learning in graph data for network embedding. We will also investigate how to operate convolutional layers on non-Euclidean graph data in Geometric Deep Learning (GDL). We focus on two different RNA representation approaches in the application of GDL. For the first approach, we design graph representations for encoding RNA, which can be used in graph convolutional networks. For the second approach, we design RNA 3D point cloud representations that can be used in deep learning methods on 3D point clouds. The key questions that arise;

- How to encode the geometry shape of the RNAs as 3D Point cloud representations?
- How to encode the geometry shape of the RNA as a graph, and the nodes consist of $multi-dimensional$ continuous features.
- How to read a graph sequentially and process non-Euclidean graph data in a meaningful and consistent order?
- How to utilise convolution operations on graphs?
- How to address a solution for arbitrary input order? In traditional CNN models, the input data has to be in the same dimension.
- How to extract important structural patterns encoded in a graph to address the graph classification problem?

The first GDL method we use in our RNA application is the deep graph convolutional neural network (DGCNN) [22]. DGCNN is an end-to-end deep learning method and proposes a convolutional operation on the spatial neighbours. DGCNN's localised convolutional approach is related to two graph kernel methods (WL subtree kernel [18] and Propagation Kernel [151]). The method also proposes a new SortPooling layer. This layer sorts the nodes of the graphs in a meaningful order and resizes the input graphs to use in traditional CNN architectures. This method outperforms graph kernel methods in particular graph representations in our RNA applications. The second GDL method we use is the Graph Isomorphism Network (GIN [25]. In GIN, the understanding of the representation properties is a challenging problem. GIN [25] express a new theoretical framework that able to use different graph structures. First, GIN transforms input data into vector representations, and then uses machine learning methods for these vector representations. The method uses methods similar to the message-passing algorithm to aggregate information from the node neighbours. The third method we use is structure2vec [24]. structure2vec, similar to the WL method, iteratively updates nodes according to information aggregated from the neighbours. The fourth method we use in our RNA application is Graph U-Nets (gUNets) [28]. The method proposes upsampling (gUnpool) and downsampling (gPool) pooling operation. Finally, we apply the LCGNN$_{GIN}$ [29] method. LCGNN$_{GIN}$ is an extension of the GIN method. We will explain all these methods in the next section.

We also design RNA 3D point cloud representations to use in deep learning methods on 3D point clouds. The input for these deep learning methods is unordered point sets. Three coordinates (x, y, z) information and additional representations encoded for a set of 3D points. In our RNA application, the three coordinates (x, y, z) are coordinates of $C3$ atoms of each nucleobase of the RNA chains. We also add three additional channels. These channels are our three RNA representations' $1-dimensional$ node features proposed in the previous chapters. We also add zero paddings and copy paddings to make all representations the same size. The deep learning methods on 3D points we use are PointNet [33], PointNet++ [34], and PointConv [35] methods. These methods get an unordered set of points as an input and so ignores RNA sequence ordering information.

6.2 Deep Learning on Graph Neural Networks

In this section, we will explain the graph convolutional neural networks we applied to our RNA graph dataset. The methods are DGCNN [22], GIN [25], structure2vec [24], gUNets [28], and LCGNN$_{GIN}$ [29].

6.2.1 Related Work

In graph neural networks (GNNs), to extract structural features of the nodes, aggregation methods are applied. Each node's neighbour node features iteratively computed, aggregated (message passing) and transferred to the given node to build and update the given node features [25, 152, 153]. This process has an analogy with the WL subtree kernel [18] which aggregate information from the neighbours and updates the current node labels according to its neighbour information. Each feature of the node is in the form of the representation vector.

The goal of the convolution operations on the graph data is to learn node representations [28]. The computation of the nodes is similar to graph kernel methods. After computing node representation, we can perform a graph classification on the learned graph representations.

Recently, there is much research on graph neural networks. In 2017, Kipf and Welling [23] proposed the graph Laplacian method for graph convolutional networks. The method performed promising results on node classification in a graph. The forward propagation of GCNs is defined as:

$$X_{i+1} = \sigma(\tilde{D}^{\frac{-1}{2}}\tilde{A}\tilde{D}^{\frac{-1}{2}}X_iW_i) \tag{6.1}$$

Where X is the feature matrix of layer i, A is an adjacency matrix and $\tilde{A}$ include self-loops ($\tilde{A} = A + I$), $\tilde{D}$ is a diagonal node degree of the $\tilde{A}$, W is a trainable matrix and σ is the activation function. Graph U-Nets [28] (gUNets) is very similar to GCN [23], additionally, the gUNets architectures consist of gPool and gUnpool layers for network embedding.

Deep learning methods are related to graph kernel methods. For instance, DGCNN is related to the Weisfeiler-Lehman (WL) subtree kernel [18] and Propagation Kernel (PK) [151]. These two kernel methods are update node labels (vertex feature) iteratively according to their neighbours.

The traditional CNN models can only handle input data in the same sizes [154]. The fully connected layer in a CNN architecture allows a fixed size array as input for classification. The dimension of the data can be any virtually sizes for other parts of the CNN architecture (convolutional, pooling, and batch-normalisation). Therefore, it is difficult to use non-regular graph data as input. DGCNN resizes the graph size in the SortPooling layer. They remove some rows from large graphs and add zero rows to small graphs to resize the graphs. Then traditional CNN models can be employed. GIN, structure2vec, and gUNets methods have no fully connected layers. After convolutional and pooling operations they directly connected to the softmax function to minimize the loss. For graph neural network the most important thing is to learn local representations for network embeddings in the convolutional layer.

6.2.2 Deep Graph Convolutional Neural Network (DGCNN)

Deep Graph Convolutional Neural Network (DGCNN) [22] is the first deep learning method we applied to our RNA graph representations. DGCNN is designed to read non-Euclidean graph data to learn a classification function.

DGCNN first proposes a new spatial graph convolutional layer that extracts the multi-dimensional node's local substructure features and defines a consistent node order [22]. The method draws an analogy with two graph kernel methods (WL subtree [18], PK [151]). The method describes a consistent and meaningful vertex ordering. The convolutional layer is defined as follows.

$$Z^{t+1} = f(\tilde{D}^{-1}\tilde{A}Z^{t}W^{t}) \tag{6.2}$$

where A denotes adjacency matrix of the graph and $\tilde{A} = A + I$ denotes adjacency matrix with added self-loops. Z denotes the input layer ($Z^0 = X$) and X denotes the input graph's node features matrix ($X \in R^{n\times c}$). n denotes the number of vertices and c denotes the dimension of node features or number of channels in X. A column in X is its feature channel. $\tilde{D}$ denotes diagonal degree matrix ($\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$). W is a trainable filter matrix of graph convolution parameters ($W \in R^{c\times c'}$). f is a nonlinear activation function and Z^t is output of the t^{th} graph convolutional layer. Here, ZW transfers c channels to c' channels. The output of this part is a concatenation of h graph convolutional layers that such $Z_h = [Z_1, ..., Z_h]$. The graph convolutional layer is similar to the graph Laplacian method proposed by Kipf and

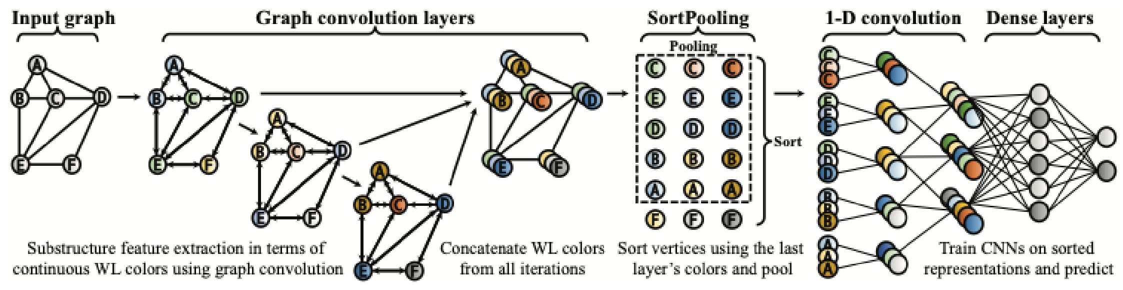

Input graph
Graph convolution layers
SortPooling
Pooling
Sort
1-D convolution
Dense layers
Substructure feature extraction in terms of continuous WL colors using graph convolution
Concatenate WL colors from all iterations
Sort vertices using the last layer's colors and pool
Train CNNs on sorted representations and predict

Fig. 6.1 DGCNN architecture. The DGCNN structure is obtained from [22]

Welling [23] (see eq: 6.1) The connection between this layer and two graph kernel methods are explained in [22].

Second, DGCNN includes a new SortPooling layer to sort node feature descriptors according to defined vertex order in the previous spatial convolutional layer. In this layer, the nodes in a graph are sorted according to their structural features in a meaningful order and allows the network to learn from the graph structure. The structural features are produced similar to the WL node label refinement process. The input of this layer is an $n \times \sum_1^h c_t$ tensor $Z^{1:h}$, and the output of this layer is $k \times \sum_1^h c_t$. Here, n denotes the number of nodes of input graphs layer, c_t denotes the number feature channel (node feature dimension), h denotes the number of graph convolution layers, and k denotes a user-defined integer.

The output of the SortPooling layer is a unified graph-size. The SortPooling layer fixes the size of the output tensor. If the size of the graph is more than a user-defined integer (k) the graph size truncated (delete $n-k$ rows if $n > k$), and if the size of the graph is less than the user-defined integer (k) or extended (add $k-n$ zero rows if $n < k$) with zero rows. The output of the SortPooling layer is unified input sizes for the next layer. The traditional 1-D convolutional and dense layer can read the sorted inputs for classification purposes. In this way, the SortPooling layer is a bridge between the graph convolutional layer and traditional CNN layers.

6.2.3 GIN

Graph Isomorphism Network [25] proposes a simple convolution operation that able to capture local feature and express a new vector representation of nodes in a graph for graph classification problems. GIN has a connection with the Weisfeiler Lehman graph isomorphism test [18] and GNNs. The idea of the WL label refinement is to, iteratively, aggregate information from the neighbour nodes' labels to update and refine the current node label. Modern GNNs update given nodes feature vector to capture features of neighbour nodes. GIN investigate and express the representational power GNNs. Similar to the WL and modern GNNs, GIN iteratively updates a node's feature vector via aggregating information from the neighbours. This method,

- first, represent a multi-set that consists of a given node's neighbours feature vectors.
- second, an aggregation function (sum, mean, max) applied to the multi-set. A multi-set is a tuple ($X = (S,m)$) that has multiple distinct (S) elements. Where m gives a multiplicity of the tuple elements ($m : S \rightarrow N \geq 1$)

In use of the GIN method, the neighbours of the given node's features are represented in a multi-set. So the aggregation function is based on the multi-set that differ from the WL method. The function is to aggregate multi-sets into different representations. The mathematical definition is as follows,

A graph $G = (V,E)$ is consist of a set of vertices (V) and edges (E) ($G_1,...,G_n \subseteq G$). X is a matrix of node features (X_v for $v \in V$), H_g is representation vector, and $Y_1,...,Y_n \subseteq Y$ is graph labels. In GNNs, the task is to learn a representation vector to determine the label of a graph for graph classification problems. GNNs iteratively use two important functions. These functions are an 'AGGREGATION' function which used for first-order adjacent node features and a 'COMBINE' function which combine aggregated features. The GNN layer is formulated as follows to update node representations,

$$a_v^{(k)} = AGGREGATE^{(k)}(\{h_u^{(k-1)} : u \in N(v)\}), h_v^{(k)} = COMBINE^{(k)}(\{h_u^{(k-1)}, a_v^{(k)}\}) \quad (6.3)$$

Here, $h_v^{(k)}$ is a feature vector, where $h_v^{(0)} = X_v$, and $N(v)$ is v's neighbours nodes. After the AGGREGATION and the COMBINE function, a READOUT function is used to aggregate all node features into a graph for graph classifications.

$$h_G = READOUT(\{h_v^{(K)} | v \in G\}) \tag{6.4}$$

GIN uses multi-layer perceptrons (MLPs). Vertices representations are updated by multi-layer perceptrons (MLPs) as,

$$h_v^k = MLP^k((1+\varepsilon^k).h_v^{(k-1)} + \sum_{u \in N_v} h_u^{(k-1)}) \tag{6.5}$$

where, ε is a fixed scalar or a learnable parameter and k denotes the k^{th} layer. The learned nodes can be directly used to produce the embedding of the entire graph. GIN proposes the following READOUT function for classification tasks.

$$h_G = CONCAT(READOUT(\{h_v^{(K)} | v \in G\}) | k = 0, 1, ..., K) \tag{6.6}$$

6.2.4 Structure2Vec

Structure2Vec is our third method, a graph embedding method, in our RNA applications that capture the node's structural identity in the model [155]. The model embeds latent variables in feature spaces, and then builds a new model for each node to learn the features spaces using embedded discriminative information. The latent variable model is a structured data point, embedded via an iterative update algorithm into feature vector spaces [24]. The latent information is close for similar vertices and the discriminative information is a nonlinear mapping. The motivation of the Structure2Vec is how to learn from latent representations on the structure of the node identity [155]. As presented in Fig. 6.2, the update formula is processed on the graph topology and updates all features vectors. The node label refinements and representations are produced in two steps as below.

- 1. initialise $\mu_i^{(0)} = 0, \forall_i$

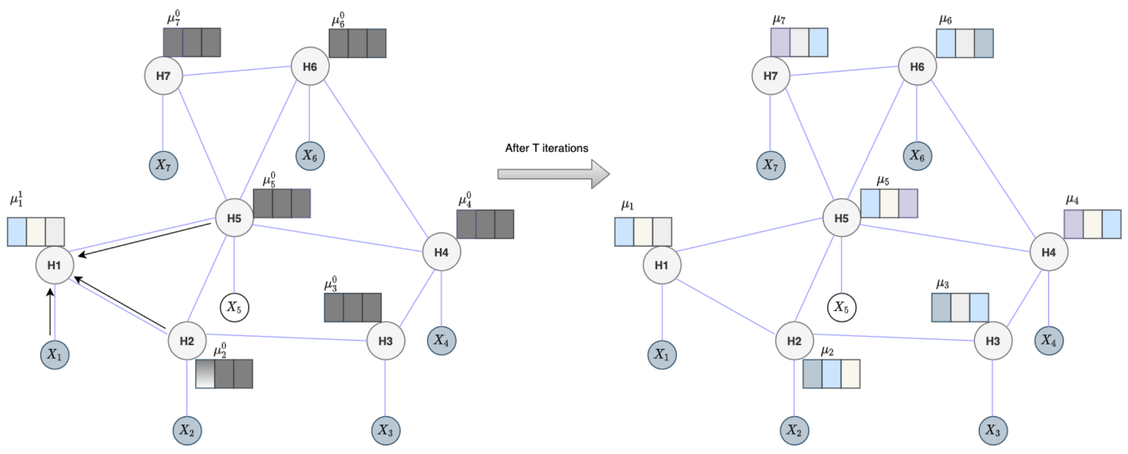


Fig. 6.2 Node representations

- 2. Iterate T times to aggregate vertex features from the node's neighbour according to the graph topology. The aim is to produce a new feature embedding for each node in the graph. At the end of the iterations, the node features consist of information aggregated from the T-hop neighbourhood and graph topology. The method updates each feature representation at each iteration as
$\mu_i^{(t)} \leftarrow \sigma(W_1 X_i + W_2 \sum_{j \in N_i} \mu_j^{(t-1)}), \forall_i$

Where W_1 and W_2 are trainable parameters, N_v is node v's neighbours set, X_i is additional feature of node v, and σ is non-linear activation function [156]. The output of this representation is connected to a soft-max layer for end-to-end graph classification.

Structure2Vec propose a new representation inspired by the WL kernel that iteratively updates node features according to the features aggregated from its neighbours. The degree of the nodes is used to find the similarities between vertices. If two nodes have the same degree in the network they are structurally similar, if their neighbour also has the same degree in case they structurally more similar [155]. Structur2Vec has an analogy with graphical model (GM) algorithms by using mean-field and belief propagation. Structure2Vec is treated each data point as a latent variable, embed GM in the feature spaces, and then define a kernel by using an inner product in the embedded space. The similarity calculated in a hierarchy. Structure2Vec aggregates information from the neighbour similar to the message passing algorithms with nonlinear function mappings in each step. The differences of the Structure2Vec to message passing algorithm are,

- At first, the structural similarity between nodes measured in a graph according to their degree
- Second, produce a multi-layered graph where each layer is weighted and compute the structural similarity of a level in a hierarchy.
- Third, to generate a context for each vertex of the multi-layer graph a mean-field update fashion is used. The sequence of nodes (context) is used to learn latent variables by standard methods. These steps of the hierarchy are recursively updated according to input graphs.

In our RNA application, this method is important because each node in the graph representation has specific properties. They consist of reach structural information. Using this method is helping to measure node similarities of the graph. In terms of both memory and computation requirements, this method is very efficient in our RNA representations.

6.2.5 Graph U-Nets

Graph U-Nets [28] proposes two pooling operations (gUnpool, gPool) and a trainable projection vector p that adapts to use a non-Euclidean graph data in a neural network. The method is based on up-sampling and down-sampling pooling layers. gPool is applied to enable down-sampling on the graph to encode high-level node features. In the gPool layer, the aim is to select a subset of nodes in a graph to build a new and smaller graph form. To do this, a trainable projection vector (p) is proposed to apply the node features. Then, the largest scalar projected nodes are received to form the smaller graph. The (p) is applied as follows. $y_i = x_i p/||p||$, where x_i is the feature vector of node i and y_i is the scalar projection of x_i on the trainable projection vector to compute the amount of information kept in the direction of the p. The aim is to preserve information from the original graph. The gPool layer i's propagation rule is processed as follows,

$$
\begin{aligned}
y &= X^l p^l / ||p^l||, \\
idx &= rank(y,k), \\
\tilde{y} &= sigmoid(y(idx)), \\
\tilde{X}^l &= X^l(idx,:), \\
A^l + 1 &= A^l(idx,idx), \\
X^{l+1} &= \tilde{X}^l \odot (\tilde{y} 1_C^T)
\end{aligned}
$$

where k is the number of selected nodes, $rank(y,k)$ returns the indices of the k largest values in y. The goal of the index selection is to preserve the order of the input graph's position information.

gUnpool is processed on the gPool layer to restore the graph into its original topology as an inverse operation. To enable the up-sampling operation, the recorded selected nodes location of the gPool layer is used to place the nodes into their original places. gUnpool is processed as follows:

$$X^{l+1} = distribute(0_{N \times C}, X^l, idx)$$

where $0_{N \times C}$ is the feature matrix of the new graph which initially was empty.

As presented in Fig6.3, the gPool layers reduce the size of the original graph to encode higher-order features. GCN layers aggregate information from each node's first-order information from the neighbourhood nodes before each pooling operation. k^{th} graph power G^k is used to avoid isolated node problem and to increase the connectivity between nodes at the maximum of k hops where $k = 2$ is used in this work.

gPool and gUnpool layers work like encoder-decoder operation. gPool layers encode higher-order node features and gUnpool layers restore the graph to its previous structure. Finally, a readout function is added to the outputs of GCN layers in the decoder part, and

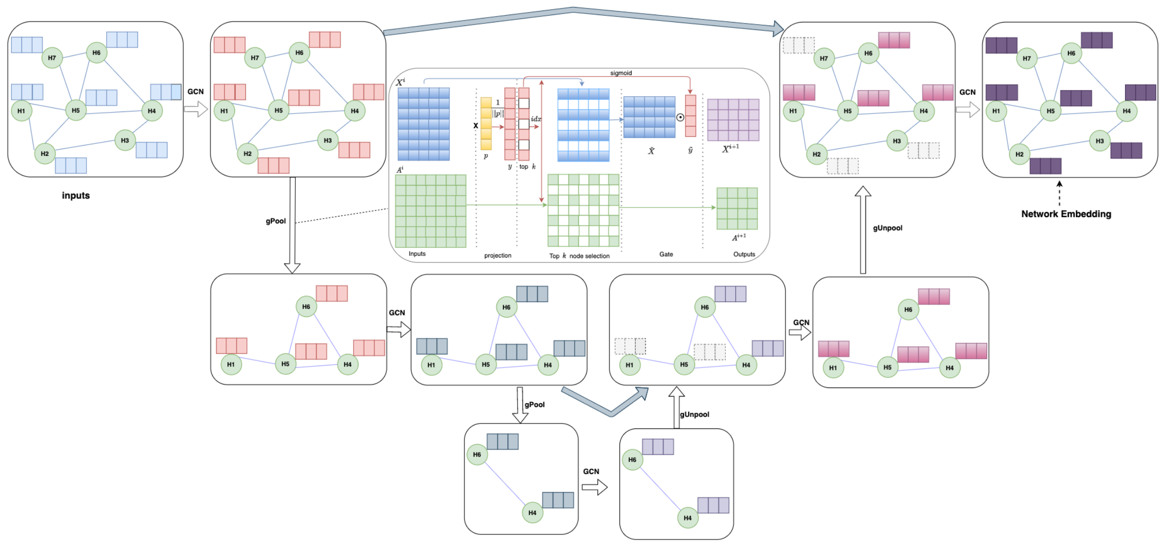


Fig. 6.3 Graph U-Nets

then resulting feature vectors are concatenated and fed into a two-layer feed-forward network for predictions [157].

6.2.6 LCGNN

Label Contrastive Coding based Graph Neural Network (LCGNN) [29] aims to use graph label information more comprehensively. It focuses on the output layer of the model for classification. First, LCGNN introduce a novel label contrastive coding that learns to discriminate different class labels.

Second, LCGNN use a batch of labelled graphs as input (G_b). In each mini-batch iteration, the set of key and query graphs are the same as G_b. The graph encoders (f_k and f_q) initialized with the same parameter ($\theta_q = \theta_k$). LCGNN uses two graph encoders (G_q and G_k) to encode them into low dimensional representations (f_q, f_k) where $q = f_q(G_q)$, and $k = f_k(G_k)$, f_q denotes key graphs, and f_k denotes query graphs. Both query and key graphs have the same topology. In our application, we use GIN as a graph encoder. GIN explained in section 6.2.3 in this chapter.

Third, LCGNN propose a dynamic label memory bank and a momentum updated encoder as a comprehensive dictionary lookup task. The dynamic label memory bank consists of the key graph representations and their labels to compute label contrastive loss. The dictionary lookup task computes label contrastive loss and updates the label-known graph classification. LCGNN uses multiple instances to pull the same label instances closer and to push the different label instances away. Thus, the label contrastive coding treats the same label as similar pair and the different label as dissimilar. The size of the memory bank and the size of the labelled graphs are equal. The memory bank contains encoded graph representation and its label (k, y). During the training the encoded key graphs updated.

Finally, LCGNN proposes a contrastive loss to extract discriminative information for the model performance. The model uses a mixed loss (combine contrastive loss and classification loss) to train graph encoder (f_q) and graph classifier more efficiently.

$$L_{total} = L_{Cla} + \beta L_{LC} \tag{6.7}$$

Here, β controls the relative weights between both loss, L_{LC} denotes contrastive loss, and L_{Cla} denotes classification loss. Then graph encoder and classifiers are updated according to L_{total}.

6.3 Deep Learning on 3D Point Clouds

Traditional deep learning methods accept regular data formats as input. These formats are in the grid (voxel), 2D image, mesh, and vector features. For instance, for image classification, 3D images converted to 2D images and then processed in a traditional CNN architecture. The characteristic pieces of information are lost during this transformation.

Point clouds are geometric point sets in a metric space. Deep learning approaches on the 3D point cloud use geometric data type as input. The input data uses three coordinates (x, y, z) information and additional channels. In this section, we extract three coordinates (x, y, z) information of the $C3$ atom of each nucleobase of the RNA strands, and then use them in deep learning applications in 3D Point clouds. We also add additional channels to our 3D RNA point cloud representations. The additional channels are our proposed representations introduced in the previous chapters and this chapter. Thus, each point of our RNA data has six dimensions. The first three dimensions are three coordinates (x, y, z) of the $C3$ atom of each nucleobase in an RNA chain, and the second three dimensions are $1-\textit{dimensional}$ node features RNA graph representations (RNA_R, RNA_ABC, RNA_type). The methods we use in our RNA 3D point cloud representations are PointNet [33], PointNet++ [34], and PointConv [35].

6.3.1 Related Work

3D point clouds are a type of geometric data and consist of unordered point sets and occur in many applications such as bioinformatics, autonomous driving, and robotics [158–160]. Traditional methods, specifically convolutional neural networks, allow using the same trainable filters on regular data representations. A fully connected layer only accepts the same dimensional data as input. It is difficult to apply convolutions operations to the unordered 3D point sets.

Deep learning methods have a range of approaches [33–35, 158, 161–166] on 3D data in the use of learning applications. The first approaches of 3D CNN networks [159, 160, 167] are to transform irregular 3D point clouds data to 2D regular data and then apply 2D CNN for classification. During these transformations, lots of geometric features of the data are lost.

The second approaches are based on converting unordered 3D point clouds data to regular 3D voxels and then apply 3D convolutional networks [163–166]. These volumetric grid representations constrained by their volumetric resolutions [33, 35] and require too much memory. These methods are computationally expensive and difficult to process on large point clouds.

The third approaches are on point-based networks. These methods designed to take unordered 3D point clouds as input. The most pioneering method among them is PointNet [33] which only practicable on global features, and it has no solution to capture the local context of points. PointNet++ [34] is an upgraded version of the PointNet. PointNet++ uses local point context and sample group and apply a PointNet to local features as a mini PointNet. Then, it proceeds to extract features from all points, combining smaller units to form larger groups. Both PointNet and PointNet++ use max-pooling to aggregate features from different points. PointNet inspired by plenty of research. It is a new de facto standard on 3D point clouds [158].

6.3.2 PointNet

The PointNet method [33] uses three coordinates (x, y, z) of points and additional dimensions as inputs. 3D point sets are not required to be in a specific order. The PointNet method is designed to addresses a solution for unordered point clouds in 3D. PointNet proposes a model that can be trained on 3D shapes for classification where the subset points are meaningful and in Euclidean space. Thus, the network model can work on local features of nearby points by using a distance metric.

The most important feature of the network is to use max-pooling as a single symmetric function to aggregate 3D information of all the points of unordered input. Using the max-pooling layer is helpful to keep responses with the highest activation for comparison to all members. In all comparison, the largest pool stays and others discard. Therefore, the order of the members does not change the maximum value. To aggregate the information, the sum of the points does not depend on the order of the points, and it does not affect the results.

$$f(\{x_1, ..., x_n\}) \approx g(h(x_1), ..., h(x_n)) \tag{6.8}$$

where $f : 2^{R^N} \rightarrow R$, $h : R^N \rightarrow R^K$, and $g : R^K \times ... \times R^K \rightarrow R$ is a symmetric function [33]. Here, h treated as a multi-layer perceptron and g as a max-pooling function. The goal is to learn a set of f's to capture different properties by using a collection of h.

Max-pooling also selects interesting local and global features of the whole point cloud and combine them to solve the interaction problem between points. The network model learns a set of optimization functions, which are used to choose informative points of the 3D point sets [33]. As presented in Fig.6.1, the output of the max-pooling layer is the input of the global feature. Finally, a fully connected layer aggregates all learnt optimized values and is used to address the classification problem. Each shape of the global feature is a vector

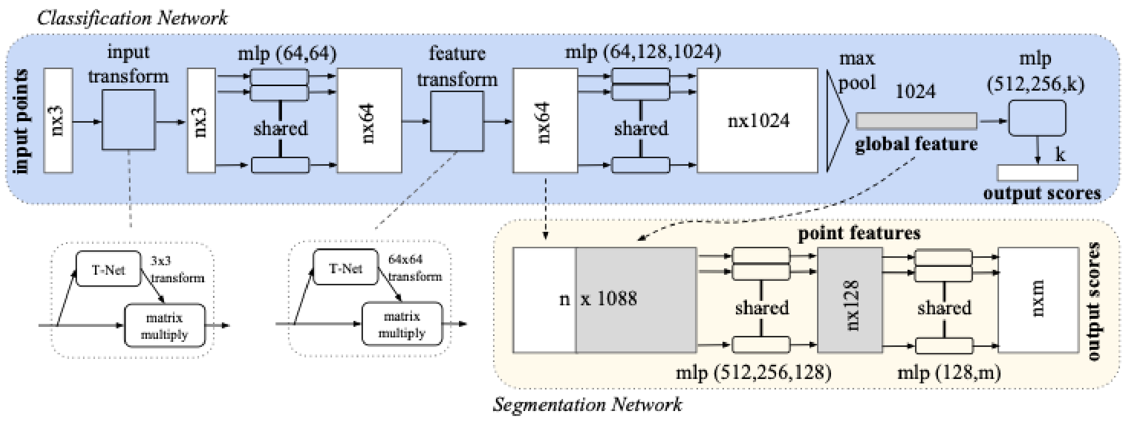


Fig. 6.4 Pointnet architecture. The figure is obtained from [33]

$[f_1,...,f_K]$ that can be trained in machine learning classifiers and multi-layer perceptron classifiers.

The method uses two joint alignment networks to make point cloud invariant during transformations. These networks are T-net transformations presented in Fig.6.4. These two mini-networks can directly apply to the input points to confine the feature transformation matrix to be close to the orthogonal matrix [33]. Thus,input information is preserved. Using orthogonal transformation does not change the input information. A feature alignment matrix (A) is predicted by T-net and used for regularization to get better performance during the classification.

$$L_{reg} = ||I - AA^T||_F^2 \tag{6.9}$$

PointNet provides a unified architecture and that applicable to our RNA data set for classification. In our RNA application we use the blue part of the Fig.6.4 for classification.

6.3.3 PointNet++

PointNet uses full features of point sets to learn a spatial encoding of each point and use a max-pooling operation to aggregate whole individual point features to build a global feature. PointNet is not able to capture local features [34] produced by a metric. But, for the success of the convolutional operations, the local features are important. The contribution

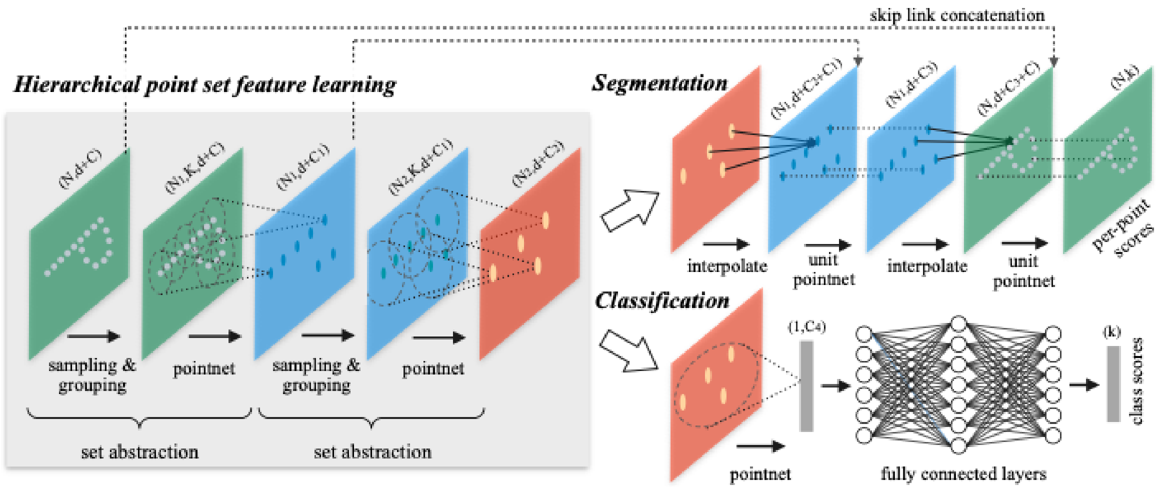


Fig. 6.5 Pointnet++ architecture. The figure received from [34]

of PointNet++ to PointNet is to add a hierarchical network structure to learn local features. The PointNet++ recursively applies PointNet to local patterns of group point sets where the points are in Euclidean space and treated hierarchically.

Pointnet++, first, divides the set of geometric points into local regions, then captures geometric features from neighbours of these small units in Euclidean space. These mini local regions are then grouped to produce larger partitions with fewer members. With this hierarchy process, higher-level features are produced, and the process repeated until whole points features captured.

As presented in Fig 6.5, in the set abstraction level, the PointNet++ uses the hierarchical point set feature learning to extract multiple local scales at each level and then combine them. PointNet++ has three key layers: sampling, grouping, and PointNet [34]. The sampling layer uses an iterative farthest point to select subset points from the input points $(N, d+C))$ that describe the centroids of the local regions. Here, N is the number of input points, d is the dimension of the three coordinates (ex: x, y, z), and C is the dimension of point features. The input $(N, d+C)$ of the grouping layer is a set of points. The output $(N', K, d+C)$ of this layer is a group of point sets. The point sets are local regions. Here, K is the number of 'neighbouring' of centroid points in a radius. The ball query finds points within a radius. The grouping layer constructs sets of local regions around the centroids. The output $(N', K, d+C)$ of the grouping layer is the input of PointNet layer. Here, a mini-PointNet is used to encode all local features into a higher-level feature vector. The output $(N', d+C1)$ of this layer is the input of the second set abstraction level. Here, N' is subsampled points or local regions.

6.3.4 PointConv

PointConv [35] introduces a new dynamic filter to a new convolution operation for building a deep convolutional architecture to use in 3D point clouds. The method uses weight and density functions to treat the convolution operation as a nonlinear function on local regions of 3D point clouds. The weight functions are learned with MLP layers, and the density functions are learned with kernel density estimations and a MLP layer. PointConv proposes a novel reformulation, a density re-weighted convolution, that changes the summation order to reduce memory consumption. PointConv defines a novel permutation-invariant convolution operation as follows.

$$PointConv(S,W,F)_{xyz} = \sum_{(\delta_x,\delta_y,\delta_z)\in G} S(\delta_x,\delta_y,\delta_z)W(\delta_x,\delta_y,\delta_z)F(x+\delta_x, y+\delta_y, z+\delta_z) \tag{6.10}$$

Where $(\delta_x,\delta_y,\delta_z)$ is any position of the point local region G, $S(\delta_x,\delta_y,\delta_z)$ is the inverse density at point $(\delta_x,\delta_y,\delta_z)$, $F(x+\delta_x, y+\delta_y, z+\delta_z)$ is the feature of a point in G at $p = (x,y,z)$. The idea is to use MLPs to approximate the weight function using three coordinate information and the S function. Thus, PointConv approximate learns weights for convolution. Kernel density estimation is used to compute the inverse density scale (S). PointConv uses a hierarchical structure,sampling, grouping, and the PointConv layer, similar to PointNet ++ [34].

PointConv reformulates the convolution operation to reduce memory consumption by using matrix multiplication and $2d$ convolution defined in the following formula.

$$F_{out} = Conv_{1x1}(H, (S.F_{in})^T \otimes M) \tag{6.11}$$

M is an input of the last layer of MLP, H is the weight of the last layer in MLP, and $Conv_{1x1}$ is $1x1$ convolution. The proof of the Eq. 6.11 explained in [35].

6.4 RNA Representations for Geometric Deep Learning

In this section, we provide various representations for encoding tertiary RNA structure to find most effective representation to classify RNA molecules in the use of geometric deep learning (GDL). First, we propose RNA graph representations consist of multi-dimensional node features because deep learning on graphs can use multi dimensional node labels as input. Second, we propose RNA 3D point cloud representations consist of three coordinate information of $C3$ atom (see. 5.3 and 5.4), and three additional channels where each channel is a node features of our RNA graph representation. The encoding methods are base on (x,y,z) coordinate of $C3$ atom of nucleobases, sequence of nucleobases, `'arc length'`, `'curvature'`, and `'base position'`.

6.4.1 RNA Graph Representations for to use in Graph Neural Networks

In this representations, (A,B,C) refers to arc-length (A), base position (B),and curvature (C) respectively and explained in Chapter 5. Each representation consist of m distinct node labels.

- RNA_ABC-II: In this representation, first, we separately applied `'arc length'` and `'curvature'` to three coordinate (x,y,z) position of the $C3$ atom of each nucleotide. Then, we find a min distance between a nucleotide and its nearest neighbour nucleotide. To do this, we used the distance between each $C3$ atom of each nucleobase using three coordinate information of $C3$, where each nucleobase contains only one C3 atom. Second, we build $(n \times 3)$ matrix where the first column is `'arc length'`, the second column is min-distance, and the third column is `'curvature'`. Therefore, each node in a graph has three node features. The first features obtained by applying `'arc` length', the second features obtained by applying min distance (`'base position'`), and the third features obtained by applying `'curvature'` to the $C3$ atom. There is no clustering on node labels. The dimension of nodes in each graph of this representation is $(n \times 3)$.

- RNA_AC-II: In this representation, first, we separately applied `'arc length'` and `'curvature'` to three coordinate (x, y, z) position of the $C3$ atom of the nucleotides. Second, we build $(n \times 2)$ matrix where the first column is `'arc length'`, second column is `'curvature'`. Therefore, each node in a graph has two node features. The first features obtained by applying `'arc length'`, the second features obtained by applying `'curvature'` to the three coordinate information of the $C3$ atom. There is no clustering on node labels. The dimension of nodes in each graph of this representation is $(n \times 2)$.
- RNA_C-II: In this representation, we applied `'curvature'` to three coordinate (x, y, z) position of the $C3$ atom of the nucleotides to build $(n \times 1)$. matrix. Each node in a graph has one node features. There is no clustering on node labels.
- RNA_A-II: In this representation, we applied `'arc length'` to three coordinate (x, y, z) position of the $C3$ atom of the nucleotides to build $(n \times 1)$ matrix. Each node in a graph has one node features. There is no clustering on node labels.
- RNA_XYZ This representation includes three (x, y, z) coordinates of the $C3$ atom. There is no clustering on node labels. The dimension of nodes in each graph of this representation is $(n \times 3)$.

6.4.2 RNA 3D Point Cloud Representations for Deep Learning on 3D Point Clouds

In this section, we propose RNA 3D point cloud representations. The representation includes three coordinates (x, y, z) of the $C3$ and three additional features. The dimension of each data is (4298×6). 4298 refer to the number of nucleobase of the longest RNA chain in our

RNA data set. The additional channels are (A, B, C) refers to `'arc length'` (A), `'base position'` (B), and `'curvature'` (C) respectively. To unify the size of the data files, we add zero paddings and copy paddings. The representations are,

- RNA_1_Z: This representation includes three coordinates (x,y,z) information of the $C3$ atom, and node features of the RNA_ABC-II representation introduced in the previous section, where node features consists of $3-dimensional$ continuous features. Zero paddings are applied to unify the size of all data.

- RNA_1_C: This representation is similar to RNA_1_Z. Here, copy paddings are applied to unify the size of all data.

- RNA_2_C: This representation includes a concatenation of three coordinates (x,y,z) of the $C3$ atom, sequence of the RNA nucleobase, $1-dimensional$ node features of the RNA_ABC (see section 5.3), and $1-dimensional$ node features of the RNA_type (see section 5.3). Copy paddings are applied to unify the size of all data.

6.5 A General Framework of Dataset and Representations

In chapter 3, we presented a novel RNA graph classification data set. In Chapter 4 and Chapter 5, we introduced compact RNA representations for encoding 2D/3D RNA structures in the graph-structured form. In this chapter, we proposed additional 3D RNA representations for use in Geometric Deep Learning.

In this section, we provide a framework of the data set and encoding methods. The framework aims to illustrate the whole process of the preparation of the RNA data set and RNA representations. As illustrated in Fig. 6.6, at first, we collected RNA data in PDB file formats. Then, we separated them by their chains and label them from scratch according to their biological functions. Second, we extracted a sequence of the RNA bases, the secondary RNA structure, and three atomic coordinates (x,y,z) of the RNA nucleobases from the PDB files. Third, we encode each RNA structure as a graph. We also represented RNA 3D point cloud representations.

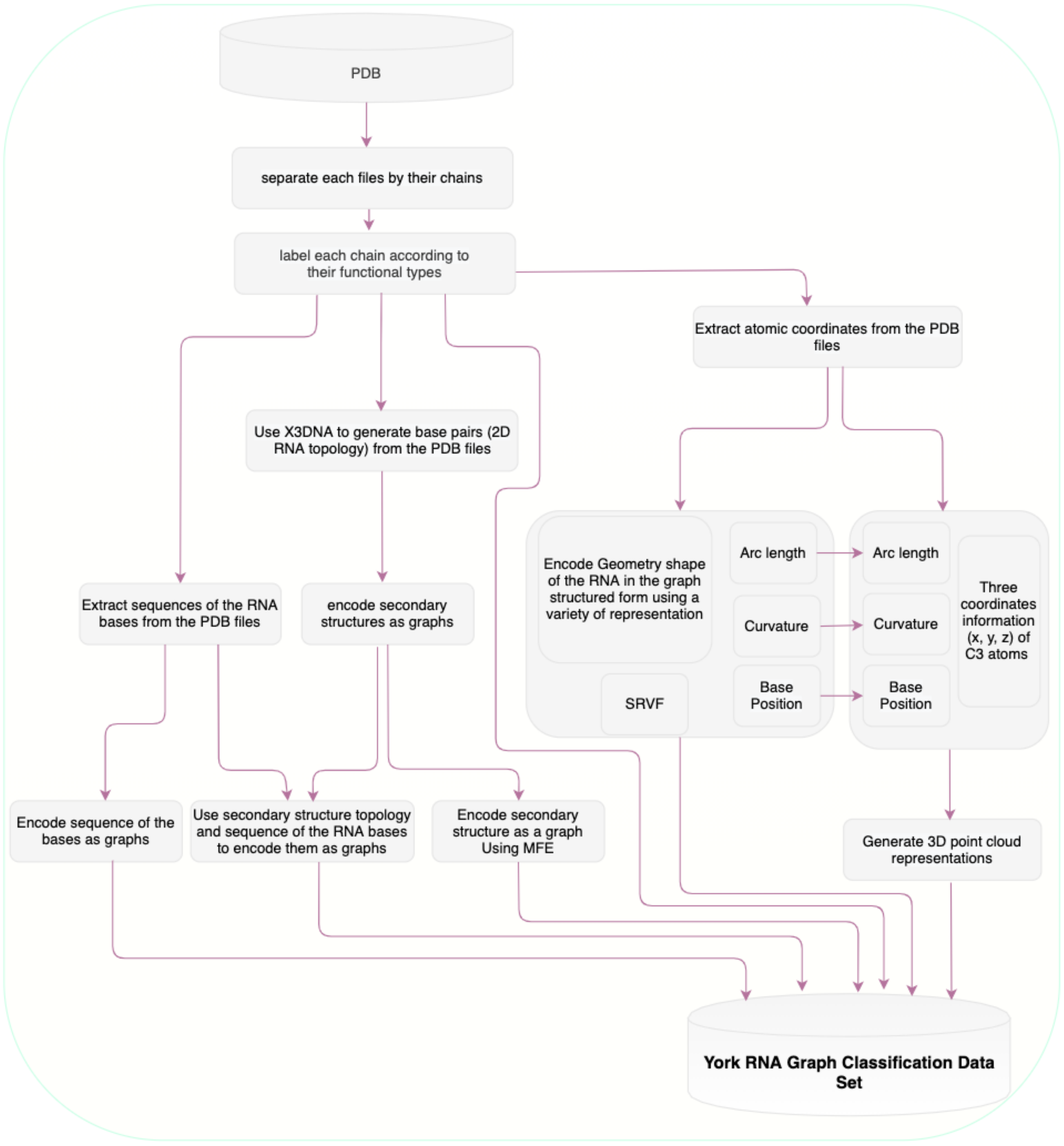

PDB
separate each files by their chains
label each chain according to their functional types
Extract atomic coordinates from the PDB files
Use X3DNA to generate base pairs (2D RNA topology) from the PDB files
Encode Geometry shape of the RNA in the graph structured form using a variety of representation
Arc length
Curvature
Base Position
SRVF
Arc length
Curvature
Base Position
Three coordinates information (x, y, z) of C3 atoms
Extract sequences of the RNA bases from the PDB files
encode secondary structures as graphs
Encode sequence of the bases as graphs
Use secondary structure topology and sequence of the RNA bases to encode them as graphs
Encode secondary structure as a graph Using MFE
Generate 3D point cloud representations
York RNA Graph Classification Data Set

Fig. 6.6 A Framework of the York RNA graph classification data set

In chapter 3, we provide a new and large RNA graph classification data set. The data set is including 3178 standalone RNA strands in PDB files. As presented in Fig 6.6, first, all PDB files were labelled from scratch by their reported functional types into 8 RNA categories. The detail of each functional class of RNA is explained in chapter 3. Second, the sequence information and three coordinates (x, y, z) of the $C3$ atom of nucleotide information extracted from the PDB files. Third, we used X3DNA to find base pairs to generate secondary structures. In chapter 4, We had a compact graph representation based on the sequence of the MFE and some refinements on the straight/unpaired part of the RNA strands to encode 2D RNA structure. In chapter 4, we introduced novel graph representations for encoding the geometry shape of the RNA in the graph-structured form. We have used Arc length, Curvature, SRVF and base pairs to encode information extracted from the geometry shape of the RNA. In this chapter, we proposed novel RNA graph representations where node consist of (1-2-3)-dimensional continuous features. We also introduced RNA 3D point cloud representations. Finally, we have collected all representation in a new folder.

Thus, the York RNA graph classification data set consist of 3D point cloud representations, primary RNA graph representation and all proposed secondary and tertiary RNA structure graph representations. The data set also includes PDB files to provide the research community with a raw data set that aid the research community in investigating new representations to solve RNA classification problems.

6.6 Results and Discussion

In this section, we provide our experimental results obtained from the proposed RNA representations. Geometric deep learning (GDL) methods are applied to classify RNA molecules according to their functions. We used two GDL approaches, deep learning methods on graphs and deep learning method on 3D point clouds, for classification purposes.

In Chapter 3-4-5, we introduced RNA graph representations to use in graph kernel methods where the RNA representations include 1-dimensional distinct node features. In this chapter, we proposed RNA graph representations include $(1-2-3)-dimensional$

	k=0.6 (77)	k= 0.75 (122)	k = 0.8 (158)	k = 0.9 (1542)
DGCNN	92.744	91.798	94.006	93.063

Table 6.1 The graphs fixed in the same size in the SortPooling layer. The numbers in the brackets are the size of the graphs after fixed. The classification results obtained from RNA_R graph representation using the DGCNN method.

continuous node features to use in GDL methods on graph neural networks. We also applied graph neural networks to all RNA graph representations introduced in the previous chapters. We also introduced RNA 3D point cloud representations to use in GDL methods on 3D point clouds.

The purpose of these RNA representations is to investigate whether the proposed representations are useful to classify RNA molecules efficiently, both in the GDL application on the graph neural networks and on 3D point clouds. The applied graph neural network methods are DGCNN [22], GIN [25], gUNets [28], Structure2Vec [24], and LCGNN$_{GIN}$ [29]. The applied 3D point cloud methods are PointNet [33], PointNet ++ [34], and PointConv [35].

The GDL methods are implemented in Python with Pytorch. The Adam optimizer used for gradient descent in all models. We trained each model up to 600 epochs with 10-fold cross-validation, and we record the best test results. After multiple trials for each method to receive the best results, we obtained the given hyperparameters.

In our DGCNN model, we use five convolutional layers with $[16,32,32,16,1]$ output channels and, in the SortPooling layer, we choose $k = 0.8$ that 80% of graphs have less than k vertices. We choose a learning rate of 0.0001. Other parts are the same as the original DGCNN model. As presented in Table 6.1, we received the best results when the graphs fixed to 158 nodes.

In our GIN model, we use GIN with the ε parameter in Equation (6.5). We use the sum AGGREGATOR function to learn node embedding. We used 4 convolutional layers, a learning rate of 0.005, and a dropout ratio of 0.4. Other parts of our model are the same as the original GIN model.

In our Strucure2Vec model, we obtained the best results when using all hyperparameters the same as the original Strucure2Vec model.

In our gUNets model, we used 4 gPool layers, and we applied gPool rates of $[0.8, 0.8, 0.8, 0.8]$. We use a learning rate of 0.004, a hidden dimension of 128, a layer dimension of 32, and a batch size of 32. Other parameters are the same as the original gUNets model.

The average of the loss function is defined as a cost function for the Geometric Deep Learning (GDL). The loss functions used to minimize the loss during training are as follows. The DGCNN, Structure2Vec, gUNets, PointConv and PointNet++ methods used Negative Log-Likelihood Loss to calculate the cost function of the classification problems. The GIN method used Cross-Entropy Loss to minimize loss during training. The $LCGNN_{GIN}$ method used a mixed loss (contrastive loss + classification loss) to train the model (see equation: 6.7). The PointNet method used the Negative Log-Likelihood Loss and a regularization loss. The regularization loss aims to constrain the feature transformation matrix to be close to the orthogonal matrix[33]: $L_{reg} = ||I - AA^T||_F^2$ where A is the feature alignment matrix estimated by a mini-network [33]. A detailed description of each cost function is available in the referenced papers. In the experiments, the cost functions used for GDL converged smoothly.

Four of our methods are inspired by Graph Kernel methods. For instance, DGCNN has a convolutional approach similar to the WL subtree kernel and PK. Structure2Vec has a very similar approach to the WL method. The WL method uses a label refinement process on the graph nodes, the refinement process is based on neighbour nodes labels. Structure2Vec uses a similar method. Here, structure2vec use the degree of the neighbour nodes to find similarities between nodes in a graph. GIN also has a similar approach to the WL method. GIN collects information from the neighbours similar to the message passing algorithm to learn a representation for the network embedding. LCGNN$_{GIN}$ is an extension to the GIN method.

In our extensive experiments, we present RNA graph representations and RNA 3D point cloud representations to use in GDL methods to determine the biological function of the RNA strands. Our data set is labelled, and the used methods based on supervised learning

	RNA_A-II	RNA_AC-II	RNA_C-II	RNA_XYZ
DGCNN	**83.596**	**80.757**	**82.019**	**84.277**
GIN	62.264	64.151	64.151	66.038
S2V	81.073	...	82.650	...
gUNets	72.642	67.610	73.899	68.553

Table 6.2 To classify the RNA dataset using DGCNN, GIN, S2V, LCGNN, gUNet and our introduced variety of representations. The representations above includes $(1-2-3)-$ *dimensional* continuous node features.

algorithms for classification purposes. As presented in Table 6.2, we applied deep learning methods to RNA representations, which consist of $(1-2-3)-dimensional$ continuous node features. The representations in Table 6.2, RNA_A and RNA_C consist of $1-dimensional$, RNA_AC consist of $2-dimensional$, and RNA_XYZ consist of $3-dimensional$ continuous node features.

The classification results of four deep learning methods are presented in Table 6.2. Empirically, in our experiments, we received the best test accuracy results by using the DGCNN method. The DGCNN method has superiority to other methods in applications of all RNA representations in Table 6.2. We obtained the best result using RNA_XYZ graph representation with an accuracy of 84.277.

Then, we applied graph neural network methods to RNA representations introduced in the previous chapter where the node labels clustered and RNA representations consist of $1-dimensional$ distinct node features. We also compare graph kernel methods and graph neural network methods. As presented in table 6.3, in all RNA graph representations, we obtained the best result by applying the Structure2Vec method to RNA_R with an accuracy of 95.3. Therefore, the most effective source of information is RNA_R obtained by the Structure2Vec method. In our investigations, for applying graph-based methods to RNA representations, the RNA_R produce better results than any RNA representations for classification purposes.

In the paper [25], GIN claim that the assertion of the GIN method is at most as powerful as the WL in graph classification problems. However, in our experiments, as the results presented in Table 6.3, the GIN method never surpassed the WL methods. In our extensive

	RNA_A	RNA_B	RNA_C	RNA_SRVF	RNA_SRVF-P	RNA_R	RNA_type	RNA_ABC	RNA_AB	RNA_AC	RNA_BC
DGCNN	**86.1**	87.4	**84.9**	81.7	82.6	94.0	85.8	87.1	88.0	**84.2**	85.5
S2V	82.3	86.7	84.5	75.3	79.5	**95.3**	86.8	**90.2**	88.0	80.1	85.2
GIN	67.29	84.9	80.8	72.6	76.7	91.2	77.4	85.2	86.8	67.6	86.2
LCGNN$_{GIN}$	72.64	86.2	83.3	73.0	76.1	93.1	78.0	85.8	85.2	74.5	86.2
gUNets	74.21	86.5	79.9	73.0	74.8	89.9	75.5	86.5	85.8	74.2	85.2
WL-OA	78.7	86.9	85.7	81.6	81.6	92.4	**87.1**	87.4	87.4	82.4	87.4
APC	76.1	84.6	82.9	80.1	81.5	89.9	85.5	86.2	85.3	76.1	85.9
SP	81.9	**87.8**	**84.9**	**82.9**	**83.4**	91.1	86.7	87.5	**88.1**	82.7	**87.7**

Table 6.3 To classify the RNA dataset using deep learning method (DGCNN, GIN, S2V, LCGNN$_{GIN}$, G-U-Net) and graph kernel methods (WL-OA, SP, APC) and our proposed variety of representations. The representations above include $1-dimensional$ distinct node features. RNA_R is (sequence + topology) of the RNA explained in Section 3.3 of this thesis.

RNA_2_C	512	1024	2048	4096
PointNet	74.465	74.513	71.759	75.739
PointNet ++	77.964	78.890	76.762	78.975

Table 6.4 The results obtained from different number of point as input on RNA_2_C 3D point cloud representations using PointNet and PointNet++.

experiments, graph kernel methods performed the best accuracy results on 6 out of 11 representations. LCGNN$_{GIN}$ method has an extension to the GIN method by adding two graph encoder (key and query graphs). In our experiments, we observed that using the two encryption method which proposed by the LCGNN$_{GIN}$ is helpful on 8 RNA graph representations.

In the comparison of the $(1-2-3)-dimensional$ continuous node features representations and clustered $1-dimensional$ distinct node representations, we observed that clustering is important and the clustered node features outperform the continuous node feature representations. This is because of the clustered nodes are previously more learned representations and include higher node features.

As the results highlighted with orange colour in Table 6.3, the *RNA_R* seems superior in all representations. Also, the ultimate performance is 95.3% and is obtained by applying S2V to *RNA_R* representation.

We represent RNA in the 3D Point clouds form to use GDL methods in 3D point clouds for RNA classification. As presented in Table 6.5, we applied 3 GDL methods in our RNA 3D point cloud representations. The methods are PointNet, PointNet++, and PointConv.

	RNA_1_C	RNA_1_Z	RNA_2_C	RNA_1_C’
PointNet	74.513	78.430	71.147	62.873
PointNet ++	75.893	70.448	78.890	63.678
PointConv	**81.604**	**82.233**	**85.220**	**82.075**

Table 6.5 The results obtained from a variety of RNA 3D point cloud representations using a variety of GDL method on 3D point clouds. RNA_1_C’ consists only three coordinates (x, y, z) information

As presented in Table 6.4, we evaluate the different number of points as input to see the classification performance. The best results obtained by using 4096 point as input in both PointNet and PointNet++.

Finally, we compare all methods on our four RNA 3D point clouds representations. We used 1024 points as input for each point-based methods. As the result presented in Table 6.5, using additional channels are helpful for classification in the use of PointNet, PointNet++, and PointConv. PointConv outperform other methods in all representations. Adding copy padding perform better than adding zero padding in the use of PointNet++. In the additional channels, the clustered nodes outperform the continuous nodes (RNA_2_C, RNA_1_C) in the use of PointNet and PointConv. We obtained the best result by using PointConv method on RNA_2_C 3D point cloud representation with 85.22% of classification accuracy.

The result obtained from the best 3D point cloud representation (RNA_2_C) higher than the all graph representations in Table 6.2 and 4 graph representation in Table 6.3 (RNA_C, RNA_SRVF, RNA_SRVF-P, RNA_AC). On the other hand the best result obtained from graph representation (RNA_R) is 10.08% higher than RNA_2_C.

We made extensive empirical experiments from different approaches, both in RNA representation methods (graph-based representations and 3D point cloud representations) and classification methods (graph kernels, GDL on graph neural networks, GDL on 3D point cloud methods). All experiments aim to determine the functional type of the RNA molecules. In our practices, for graph-based representations, we observed that the distinct node labels are more effective than continuous node labels. In the comparison of the graph kernel methods to the GDL on graph neural networks, graph neural network methods mostly perform better than graph kernel methods. In our RNA representations, graph-based methods (representations + classification methods) outperform 3D point cloud methods(representations + classification methods).

6.7 Summary

In this chapter, at first, we defined the problems of non-euclidean graph data in the use of GDL methods, and RNA 3D point cloud data in the use of GDL methods on the 3D point cloud.

Second, we explained GDL on graph neural network methods (DGCNN, GIN, LCGNN$_{GIN}$, Structure2Vec, and gUNets) and GDL methods on 3D point clouds (PointNet, PointNet++, PointConv, and three Point Transformer) for classification purposes. Third, we successfully encoded RNA in the graph-structured form where each node include $(1-2-3)-$ *dimensional* features. We also encoded and RNA in 3D point cloud representations where each data includes three coordinates of the $C3$ atom and three additional features. Fourth, we presented a framework to show all RNA representations, and we explained how we prepared all these representations. Finally, in the results and discussion section, we evaluated the performance of GDL methods on our RNA representations. We also applied graph neural network methods to our proposed RNA representations introduced in the previous chapters. We compared the results obtained from deep learning methods on graphs and the results obtained from graph kernel methods. We also made a comparison between RNA graph representations and RNA 3D point cloud representations.

In our experiments, we observed that the kernel methods are powerful methods for RNA graph classification problems. A drawback of the kernel methods does not allow to use of multi-dimensional continuous node features as input. We applied the GDL method on graphs to this kind of representations. Overall, we received the highest result from the RNA_R graph representation using a graph neural network (S2V), and also the graph neural network methods outperformed the graph kernel methods in 5 out of 11 graph representations. These methods are the most flexible methods that allow us to apply them to all proposed representations. In our all experiments on the three source of information of RNA, we received a significant result on the sequences of the nucleobases with added the topology of the 2D RNA. The result was with an accuracy of 95.3% using the Structure2Vec method.

According to our observations from the experimental results, graph kernel methods were more expensive in training time than graph neural network methods. In the use of graph kernel, there are no trainable parameters for comparing graph pairs. The output of a graph kernel method in a graph dataset is a similarity matrix. A machine learning classifier is applied to classify this similarity matrix. The classification performance depends on both graph kernel methods and machine learning classifiers. We obtained the best results from Subspace KNN when we used machine learning classifiers in our experiments. Subspace KNN is an ensemble method where the training parameters are based on Majority Vote rule and the learner type is Nearest Neighbour of 30 numbers of learners [168]. We have one or two trainable parameters for each node in GNNs. The number of parameters used in GDL methods varies with the configuration and the size of the dataset, but typically a method such as DGCNN uses in the order of a million parameters.

Finally, we also look at the power of GDL methods on the 3D Point clouds. These methods are also more flexible than traditional neural network methods. Here, the data set has to be in the same dimensional, but networks use several unordered points from the point set. Their flexibility is that the point sets do not require to be ordered. In our experiments, these networks do not perform better than graph kernel methods and graph neural network methods for RNA classification problems. Of course, the representations and approaches are different. On the other hand, we received the best result on the $1-\textit{dimensional}$ distinct node labels in classification tasks.

Chapter 7

Conclusion

7.1 Summary

This thesis aimed to find efficient representation methods to encode RNA structures into data-structured forms and use them in classification methods to determine the biological functions of RNAs. Encoding RNA structures into structured data was a challenging problem due to the complex structures of RNA strands. In the reviewed literature, the learning applications are focused on classification methods. Unfortunately, there is not much effort in the representation of the structured data. However, well-represented data increase predictive results in the use of learning methods.

In this thesis, various representation methods were introduced to encode 1D, 2D, and 3D RNA structures. The developed representation methods also combined sequence and secondary structure of the RNA, sequence and tertiary structure of the RNA, and sequence, secondary and tertiary structure of the RNA. At first, graph representation methods were introduced for RNA encoding, and then the state-of-the-art graph kernel methods were applied to RNA graph representations for comparison.

Recently, modern neural network architectures (graph neural networks) have been developed that directly consumes graph data. In this work, state-of-the-art graph neural network methods were also applied to the RNA graph representations to evaluate the classification performance. In some RNA graph representations, graph neural network methods outperformed graph kernel methods.

Furthermore, 3D Point cloud representations, which are mostly used in image classification and image segmentation, were investigated. As an additional study, at first, RNA data was transformed into 3D Point cloud representations, and then state-of-the-art 3D Point cloud methods were applied to these representations. The best classification results in RNA application were obtained from graph neural network methods. Graph kernel methods outperformed the 3D Point Cloud methods but were less efficient than graph neural network methods. In the next section, the main contributions are summarized.

7.1.1 Summary of the Main Contributions

This thesis has made contributions to bioinformatics and computer science. The contributions of the thesis are to develop a new large RNA data set, to introduce representation methods for encoding 2D/3D RNA structures, and use these representations in the classification methods, respectively.

- **Data:** The first contribution of this thesis was to prepare a new large RNA data set where RNA chains are labelled from scratch by their type of function. This data set is important for RNA classification problems. The data set consists of the 1D/2D/3D RNA structure information and their graph and 3D Point cloud representations. The aim is also to provide a platform for the research community to use in their studies. The RNA dataset is difficult to prepare. Now, if required, the research community can expand the data set with less effort. The technical details and preparation directions are explained in Chapter 3 of the thesis.

- **2D RNA Graph Representation:** The second contribution of this thesis is to encode 2D RNA structures using a sequence of the minimum free energy (MFE) on the structural components into a graph-structured form. This graph representation aims to reduce the graph size, and in contrast, receiving more information from the RNA structures to improve the classification performance. A large number of RNA strands in our data set are straight chains, and they have no secondary RNA structures. To apply classification methods to all data in the dataset, alternative representation solutions were introduced that encode all RNA strands into graph-structured form. Thus, the introduced representation can apply RNA strands of any size.

- **3D RNA Graph Representations:** The third contribution of this thesis is to investigate representation methods for encoding 3D RNA structures. In this contribution, 3D RNA structures were treated as 3-dimensional continuous curves. To encode 3D RNA structures into graph-structured from, first, three coordinates (x, y, z) information of the $C3$ atom of backbone sugar of RNA molecules were extracted from PDB files. Then, these coordinates information were then used to represent the geometric shape of RNA as a graph with geometric measures such as `'arc length'`, `'curvature'`, square root velocity function `'SRVF'`, and `'base position'`. Introduced 3D RNA graph representations tested with state-of-the-art graph kernel methods. The experiments prove that using introduced 3D RNA graph representations are help to determine the function of the RNA molecules with a high rate of accuracy.

- **Geometric Deep Learning:** Geometric deep learning methods have recently become popular, and they are powerful classification methods for machine learning tasks. The final contribution of this thesis is to analyse RNA classification methods in the use of Geometric Deep Learning applications. In this study, two different approaches of the GDL were used. The first GDL approach was based on graph neural networks. Most of the graph neural network methods can directly consume multi-dimensional continuous node featured graphs as input. Here, first, multi-dimensional continuous node featured graphs were introduced, and then state-of-the-art graph neural network methods were applied to these representations. The second GDL approach was based on 3D Point Clouds. Point clouds are geometric point sets in a metric space. Deep learning on the 3D point cloud uses geometric data type as input. The input data uses three coordinates (x, y, z) information and additional channels. For the second GDL approach, at first, RNA 3D point cloud representations were designed. Then state-of-the-art 3D Point Cloud methods were applied to the RNA 3D point cloud representations for classification purposes.

7.2 Future Directions

In this part, we discuss future directions for RNA studies. We also discuss future works that would be useful to explore other macromolecules such as enzymes and proteins.

- **RNA:** There is a strong relationship between the structure of RNA and the biological function of RNA. Predicting 3D RNA folding is important to determine its biological function. 3D RNA folding is a challenging but unsolved problem that needs to be investigated. The sequence of RNA structures folds over itself to produce secondary and tertiary RNA structures. Current studies predict secondary structures and use these secondary structures to generate predictive 3D RNA structures. It is difficult to produce the 3D Folding structure of the large RNA strands. The research community still has not found an optimal solution for 3D RNA folding problems. There is a lack of experimental data on the thermodynamic energies of 3D structures [169]. It is also challenging to obtain the 3D RNA structure data even with the latest x-ray crystallography and nuclear magnetic resonance imaging technologies [169].

- **Enzymes and Proteins:** Future developments on the representation methods will be on the 3D structure of proteins and enzymes. Proteins are types of bio-macromolecules that consist of amino acids. Structurally similar proteins have similar functions. The sequence of amino acids and the 3D structure of proteins are used to determine the type of proteins. Enzymes are special types of proteins. Enzymes are important macromolecules that catalyse almost all chemical reactions in the cell, and they are also important in the area of food science, genomics, drug discovery, and biofuels [170, 171]. Therefore, understanding the function of the enzymes would improve our understanding of life sciences. The computational methods applied to determine the type of proteins can also be applied to enzymes. The 3D structures of enzymes and proteins are similar to 3D RNA structures but have more complex structures. Our introduced 3D RNA graph representations are applicable for encoding the 3D shape of protein and enzyme structures. The representation methods introduced in Chapter 5 of this thesis for encoding RNAs can be applied to the sequence of amino acid backbones to encode proteins and enzymes in graph-structured forms.

References

[1] T. Zok, M. Antczak, M. Zurkowski, M. Popenda, J. Blazewicz, R. W. Adamiak, and M. Szachniuk, "RNApdbee 2.0: multifunctional tool for RNA structure annotation," *Nucleic Acids Research*, vol. 46, pp. W30–W35, 04 2018.

[2] I. TINOCO, O. C. UHLENBECK, and M. D. LEVINE, "Estimation of Secondary Structure in Ribonucleic Acids," *Nature*, vol. 230, pp. 362–367, 04 1971.

[3] V. Pande and L. Nilsson, "Insights into structure, dynamics and hydration of locked nucleic acid (lna) strand-based duplexes from molecular dynamics simulations," *Nucleic acids research*, vol. 36, no. 5, pp. 1508–1516, 2008.

[4] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Thirty-Second AAAI (Association for the Advancement of Artificial Intelligence) Conference on Artificial Intelligence*, 2018.

[5] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

[6] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *arXiv preprint arXiv:1706.02413*, 2017.

[7] R. C. Wilson and E. Algul, "Categorization of rna molecules using graph methods," in *Structural, Syntactic, and Statistical Pattern Recognition* (X. Bai, E. R. Hancock, T. K. Ho, R. C. Wilson, B. Biggio, and A. Robles-Kelly, eds.), (Cham), pp. 439–448, Springer International Publishing, 2018.

[8] E. Algul and R. C. Wilson, "A database and evaluation for classification of rna molecules using graph methods," in *Graph-Based Representations in Pattern Recognition* (D. Conte, J.-Y. Ramel, and P. Foggia, eds.), (Cham), pp. 78–87, Springer International Publishing, 2019.

[9] R. Dahm, "Friedrich miescher and the discovery of dna," *Developmental Biology*, vol. 278, no. 2, pp. 274 – 288, 2005.

[10] G. M. Blackburn, M. J. Gait, D. Loakes, D. M. Williams, J. A. Grasby, M. Egli, A. Flavell, S. Allen, J. Fisher, A. M. Pyle, *et al.*, *Nucleic acids in chemistry and biology*. Royal Society of Chemistry, 2006.

[11] J.-H. Spille and U. Kubitscheck, "Labelling and imaging of single endogenous messenger rna particles in vivo," *Journal of Cell Science*, vol. 128, no. 20, pp. 3695–3706, 2015.

[12] R. Carrasco-Hernandez, R. Jácome, Y. López Vidal, and S. Ponce de León, "Are RNA Viruses Candidate Agents for the Next Global Pandemic? A Review," *ILAR Journal*, vol. 58, pp. 343–358, 09 2017.

[13] A. Balcerak, A. Trebinska, R. Konopiński, M. Wakula, and E. Grzybowska, "Rna-protein interactions: Disorder, moonlighting and junk contribute to eukaryotic complexity," *Open Biology*, vol. 9, p. 190096, 06 2019.

[14] H.-Y. Huang and C.-J. Lin, "Linear and kernel classification: When to use which?," in *Proceedings of the 2016 SIAM International Conference on Data Mining*, pp. 216–224, SIAM, 2016.

[15] M. Hajiaghayi, A. Condon, and H. Hoos, "Analysis of energy-based algorithms for rna secondary structure prediction," *BMC bioinformatics*, vol. 13, p. 22, 02 2012.

[16] M. Lau and A. Ferré-D'Amaré, "Many activities, one structure: Functional plasticity of ribozyme folds," *Molecules*, vol. 21, p. 1570, 11 2016.

[17] J. Laborde, D. Robinson, A. Srivastava, E. Klassen, and J. Zhang, "Rna global alignment in the joint sequence–structure space using elastic shape analysis," *Nucleic acids research*, vol. 41, 04 2013.

[18] Y. Ding, "Statistical and bayesian approaches to rna secondary structure prediction.," *RNA*, vol. 12 3, pp. 323–31, 2006.

[19] K. Purzycka, M. Popenda, M. Szachniuk, M. Antczak, P. Lukasiak, J. Blazewicz, and R. Adamiak, "Automated 3d rna structure prediction using the rnacomposer method for riboswitches(1.)," *Methods in enzymology*, vol. 553C, pp. 3–34, 03 2015.

[20] L. Chen, G. A. Calin, and S. Zhang, "Novel insights of structure-based modeling for rna-targeted drug discovery," *Journal of Chemical Information and Modeling*, vol. 52, no. 10, pp. 2741–2753, 2012. PMID: 22947071.

[21] J. Laborde, A. Srivastava, and J. Zhang, "Structure-based rna function prediction using elastic shape analysis," in *2011 IEEE International Conference on Bioinformatics and Biomedicine*, pp. 16–21, 2011.

[22] Z. Miao and E. Westhof, "Rna structure: Advances and assessment of 3d structure prediction," *Annual Review of Biophysics*, vol. 46, no. 1, pp. 483–503, 2017. PMID: 28375730.

[23] H.-N. Lin and W.-L. Hsu, "DART: a fast and accurate RNA-seq mapper with a partitioning strategy," *Bioinformatics*, vol. 34, pp. 190–197, 09 2017.

[24] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of Molecular Biology*, vol. 43, no. 3, pp. 443–453, 1970.

[25] B. Shabash and K. C. Wiese, "Rna visualization: Relevance and the current state-of-the-art focusing on pseudoknots," *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, vol. 14, pp. 696–712, May 2017.

[26] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, “Weisfeiler-lehman graph kernels,” *J. Mach. Learn. Res.*, vol. 12, p. 2539–2561, Nov. 2011.

[27] J.-P. Vert, “The optimal assignment kernel is not positive definite,” *arXiv preprint arXiv:0801.4061*, 2008.

[28] G. Nikolentzos, G. Siglidis, and M. Vazirgiannis, “Graph kernels: A survey,” *arXiv preprint arXiv:1904.12218*, 2019.

[29] Q. Jiang and J. Ma, “A novel graph kernel on chemical compound classification,” *Journal of bioinformatics and computational biology*, vol. 16, no. 06, p. 1850026, 2018.

[30] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” *CoRR*, vol. abs/1609.02907, 2016.

[31] H. Dai, B. Dai, and L. Song, “Discriminative embeddings of latent variable models for structured data,” in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML’16, p. 2702–2711, JMLR.org, 2016.

[32] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, “How powerful are graph neural networks?,” in *arXiv preprint arXiv:1810.00826*, 2019.

[33] M. Zhang and Y. Chen, “Link prediction based on graph neural networks,” *Advances in Neural Information Processing Systems*, vol. 31, pp. 5165–5175, 2018.

[34] T. Zhao, X. Zhang, and S. Wang, “Graphsmote: Imbalanced node classification on graphs with graph neural networks,” in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pp. 833–841, 2021.

[35] H. Gao and S. Ji, “Graph u-nets,” in *international conference on machine learning*, pp. 2083–2092, PMLR, 2019.

[36] Y. Ren, J. Bai, and J. Zhang, “Label contrastive coding based graph neural network for graph classification,” *arXiv preprint arXiv:2101.05486*, 2021.

[37] Z. Chen, L. Li, and J. Bruna, “Supervised community detection with line graph neural networks.(2018),” 2018.

[38] S. S. Du, K. Hou, R. R. Salakhutdinov, B. Poczos, R. Wang, and K. Xu, “Graph neural tangent kernel: Fusing graph neural networks with graph kernels,” pp. 5723–5733, 2019.

[39] Z. Zhang, J. Bu, M. Ester, J. Zhang, C. Yao, Z. Yu, and C. Wang, “Hierarchical graph pooling with structure learning,” *arXiv preprint arXiv:1911.05954*, 2019.

[40] W. Wu, Z. Qi, and L. Fuxin, “Pointconv: Deep convolutional networks on 3d point clouds,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9621–9630, 2019.

[41] M. M. Li, K. Huang, and M. Zitnik, “Graph representation learning in biomedicine,” in *arXiv e-prints,arXiv:2104.04883*, 2021.

[42] W. Liu, A. Srivastava, and J. Zhang, “Protein structure alignment using elastic shape analysis,” in *Proceedings of the First ACM International Conference on Bioinformatics and Computational Biology*, BCB ’10, (New York, NY, USA), p. 62–70, Association for Computing Machinery, 2010.

[43] T. A. Cooper, L. Wan, and G. Dreyfuss, “Rna and disease,” *Cell*, vol. 136, no. 4, pp. 777–793, 2009.

[44] W. Saenger, *Principles of nucleic acid structure*. Springer Science & Business Media, 2013.

[45] N. Narayanaswamy, G. Suresh, U. D. Priyakumar, and T. Govindaraju, “Double zipper helical assembly of deoxyoligonucleotides: mutual templating and chiral imprinting to form hybrid dna ensembles,” *Chemical Communications*, vol. 51, no. 25, pp. 5493–5496, 2015.

[46] “Rna,” *New World Encyclopedia, http://newworldencyclopedia.org/entry/RNA, accessed 1 September 2021.*

[47] B. Wolfgang, “Rna secondary structure prediction including pseudoknots,” *University of Vienna*, 2010.

[48] “Rna class:the classification of rna,” *Cd-genomics.com, 2018, accessed 1 September 2021.*

[49] K. Srinibas, “Classification of dna: 7 criteria’s (with diagram),” *Biology Discussion*, 2018.

[50] M. Quadrini, R. Culmone, and E. Merelli, “Topological classification of rna structures via intersection graph,” in *Theory and Practice of Natural Computing* (C. Martín-Vide, R. Neruda, and M. A. Vega-Rodríguez, eds.), (Cham), pp. 203–215, Springer International Publishing, 2017.

[51] A. Rybarczyk, N. Szostak, M. Antczak, T. Zok, M. Popenda, R. Adamiak, J. Blazewicz, and M. Szachniuk, “New in silico approach to assessing rna secondary structures with non-canonical base pairs,” *BMC Bioinformatics*, vol. 16, 2015.

[52] M. Magnus, K. Kappel, R. Das, and J. Bujnicki, “Rna 3d structure prediction guided by independent folding of homologous sequences,” *BMC Bioinformatics*, vol. 20, 2019.

[53] J. Wang, J. Wang, Y. Huang, and Y. Xiao, “3drna v2.0: An updated web server for rna 3d structure prediction,” *International Journal of Molecular Sciences*, vol. 20, p. 4116, 08 2019.

[54] M. Magnus, M. Antczak, T. Zok, J. Wiedemann, P. Lukasiak, Y. Cao, J. M. Bujnicki, E. Westhof, M. Szachniuk, and Z. Miao, “RNA-Puzzles toolkit: a computational resource of RNA 3D structure benchmark datasets, structure manipulation, and evaluation tools,” *Nucleic Acids Research*, vol. 48, pp. 576–588, 12 2019.

[55] T. H. Jukes, C. R. Cantor, *et al.*, “Evolution of protein molecules,” *Mammalian protein metabolism*, vol. 3, pp. 21–132, 1969.

[56] D. Levy, R. Yoshida, and L. Pachter, “Neighbor joining with phylogenetic diversity estimates,” *arXiv preprint q-bio/0508001*, 2005.

[57] N. Kriege, F. Johansson, and C. Morris, “A survey on graph kernels,” *Applied Network Science*, 05 2020.

[58] L. Jia, B. Gaüzère, and P. Honeine, “Graph kernels based on linear patterns: Theoretical and experimental comparisons,” 2019.

[59] H. Lodhi, “Computational biology perspective: kernel methods and deep learning,” *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 4, no. 5, pp. 455–465, 2012.

[60] S. P. Dwivedi and R. S. Singh, “Error-tolerant graph matching using node contraction,” *Pattern Recognition Letters*, vol. 116, pp. 58–64, 2018.

[61] R. Raveaux, “On the unification of the graph edit distance and graph matching problems,” *Pattern Recognition Letters*, vol. 145, pp. 240–246, 2021.

[62] M. A. Eshera and K.-S. Fu, “A graph distance measure for image analysis,” *IEEE transactions on systems, man, and cybernetics*, no. 3, pp. 398–408, 1984.

[63] T. Gärtner, Q. V. Le, and A. J. Smola, “A short tour of kernel methods for graphs,” *Under Preparation*, 2006.

[64] K. M. Borgwardt and H. Kriegel, “Shortest-path kernels on graphs,” in *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005), 27-30 November 2005, Houston, Texas, USA*, pp. 74–81, 2005.

[65] K. Borgwardt, E. Ghisu, F. Llinares-López, L. O’Bray, and B. Rieck, “Graph kernels: State-of-the-art and future challenges,” *arXiv preprint arXiv:2011.03854*, 2020.

[66] K. Melnyk, S. Klus, G. Montavon, and T. O. Conrad, “Graphkke: graph kernel koopman embedding for human microbiome analysis,” *Applied Network Science*, vol. 5, no. 1, pp. 1–22, 2020.

[67] G. K. D. de Vries, “A fast approximation of the weisfeiler-lehman graph kernel for rdf data,” in *Machine Learning and Knowledge Discovery in Databases* (H. Blockeel, K. Kersting, S. Nijssen, and F. Železný, eds.), (Berlin, Heidelberg), pp. 606–621, Springer Berlin Heidelberg, 2013.

[68] B. Weisfeiler and A. Leman, “A reduction of a graph to a canonical form and an algebra arising during this reduction,” *Nauchno-Technicheskaya Informatsiya*, vol. 2, no. 9, pp. 12–16, 1968.

[69] N. M. Kriege, P.-L. Giscard, and R. C. Wilson, “On valid optimal assignment kernels and applications to graph classification,” in *Advances in Neural Information Processing Systems*, pp. 1615–1623, 2016.

[70] H. Fröhlich, J. K. Wegner, F. Sieker, and A. Zell, "Optimal assignment kernels for attributed molecular graphs," in *Proceedings of the 22nd international conference on Machine learning*, pp. 225–232, 2005.

[71] L. Hermansson, F. D. Johansson, and O. Watanabe, "Generalized shortest path kernel on graphs," in *Discovery Science* (N. Japkowicz and S. Matwin, eds.), (Cham), pp. 78–85, Springer International Publishing, 2015.

[72] P.-L. Giscard and R. C. Wilson, "The all-paths and cycles graph kernel," *arXiv preprint arXiv:1708.01410*, 2017.

[73] M. Awad and R. Khanna, *Machine Learning*, pp. 1–18. Berkeley, CA: Apress, 2015.

[74] T. K. Lee, J. H. Cho, D. S. Kwon, and S. Y. Sohn, "Global stock market investment strategies based on financial network indicators using machine learning techniques," *Expert Systems with Applications*, vol. 117, pp. 228 – 242, 2019.

[75] J. Chaudhary and J. Paulose, "Opinion mining on newspaper headlines using svm and nlp," *International Journal of Electrical and Computer Engineering*, vol. 9, p. 2152, 06 2019.

[76] S. Winters-Hilt, "Channel current cheminformatics and stochastic carrier-wave signal processing," *International Journal of Computing and Optimization*, vol. 4, pp. 115–157, 01 2017.

[77] X. Zhang and S. Liu, "RBPPred: predicting RNA-binding proteins from sequence using SVM," *Bioinformatics*, vol. 33, pp. 854–862, 12 2016.

[78] T. M. Mitchell, *Machine Learning*. USA: McGraw-Hill, Inc., 1 ed., 1997.

[79] A. Fiannaca, M. L. Rosa, L. Paglia, R. Rizzo, and A. Urso, "nrc: non-coding rna classifier based on structural features," *BioData Mining*, vol. 10, 2017.

[80] S. Singh and R. Singh, "Application of supervised machine learning algorithms for the classification of regulatory RNA riboswitches," *Briefings in Functional Genomics*, vol. 16, pp. 99–105, 04 2016.

[81] V. Tran, S. Tempel, B. Zerath, F. Zehraoui, and F. Tahi, "mirboost: Boosting support vector machines for microrna precursor classification," *RNA (New York, N.Y.)*, vol. 21, 03 2015.

[82] D. Koessler, D. Knisley, J. Knisley, and T. Haynes, "A predictive model for secondary rna structure using graph theory and a neural network," *BMC bioinformatics*, vol. 11 Suppl 6, p. S21, 10 2010.

[83] X. Wu, J. Wang, and K. Herbert, "A new kernel method for rna classification," pp. 201–208, 01 2006.

[84] Y. Yuan and Z. Bar-Joseph, "Deep learning for inferring gene relationships from single-cell expression data," *Proceedings of the National Academy of Sciences*, vol. 116, no. 52, pp. 27151–27158, 2019.

[85] G. Zararsız, D. Goksuluk, S. Korkmaz, V. Eldem, G. E. Zararsiz, I. P. Duru, and A. Ozturk, “A comprehensive simulation study on classification of rna-seq data,” *PLOS ONE*, vol. 12, pp. 1–19, 08 2017.

[86] J. Wang and X. Wu, “Kernel design for rna classification using support vector machines,” *International journal of data mining and bioinformatics*, vol. 1, pp. 57–76, 02 2006.

[87] A. Ben-Hur, C. S. Ong, S. Sonnenburg, B. Schölkopf, and G. Rätsch, “Support vector machines and kernels for computational biology,” *PLOS Computational Biology*, vol. 4, pp. 1–10, 10 2008.

[88] I. K. Nti, A. F. Adekoya, and B. A. Weyori, “A comprehensive evaluation of ensemble learning for stock-market prediction,” *Journal of Big Data*, vol. 7, no. 1, pp. 1–40, 2020.

[89] B. Ghojogh and M. Crowley, “Linear and quadratic discriminant analysis: Tutorial,” *arXiv preprint arXiv:1906.02590*, 2019.

[90] “3dna: a suite of software programs for the analysis, rebuilding and visualization of 3-dimensional nucleic acid structures,” *https://x3dna.org*.

[91] M. S. WATERMAN, “Secondary structure of single-stranded nucleic acids,” *Studies in Foundations and Combinatorics Advances in Mathematics Supplementary Studies*, vol. 1, pp. 167–212, 1978.

[92] D. Fera, N. Kim, N. Shiffeldrim, J. Zorn, U. Laserson, H. H. Gan, and T. Schlick, “Rag: Rna-as-graphs web resource,” *BMC Bioinformatic*, vol. 5, 07 2004.

[93] N. Kim, K. N. Fuhr, and T. Schlick, *Graph Applications to RNA Structure and Function*, pp. 23–51. New York, NY: Springer New York, 2013.

[94] D. Knisley, J. Knisley, C. Ross, and A. Rockney, “Classifying multigraph models of secondary rna structure using graph-theoretic descriptors,” *ISRN Bioinformatics, International Scholarly Research Network*, 11 2012.

[95] “Protein data bank japan,” *pdbj.org, accessed 1 September 2021.*

[96] “Nucleic acid database (ndb),” *Ndbserver.rutgers.edu, accessed 1 September 2021.*

[97] “Rcsb pdb,” *rcsb.org, accessed 1 September 2021.*

[98] “What is fasta format?,” *University of Michigan, https://zhanggroup.org/FASTA/, accessed 1 September 2021.*

[99] J. M. Shelton and S. J. Brown, “Fasta-o-matic: a tool to sanity check and if needed reformat fasta files,” *bioRxiv*, 2015.

[100] C. Camacho, G. Coulouris, V. Avagyan, N. Ma, J. Papadopoulos, K. Bealer, and T. Madden, “Blast+: architecture and applications. bmc bioinformatics 10:421,” *BMC bioinformatics*, vol. 10, p. 421, 12 2009.

[101] A. Dobin, C. A. Davis, F. Schlesinger, J. Drenkow, C. Zaleski, S. Jha, P. Batut, M. Chaisson, and T. R. Gingeras, "STAR: ultrafast universal RNA-seq aligner," *Bioinformatics*, vol. 29, pp. 15–21, 10 2012.

[102] A. Zielezinski, S. Vinga, J. Almeida, and W. M. Karlowski, "Alignment-free sequence comparison: benefits, applications, and tools," *Genome biology*, vol. 18, no. 1, p. 186, 2017.

[103] "Protein data bank contents guide: Atomic coordinate entry format description," *Wwpdb.org, accessed 1 September 2021.*

[104] P. S. Klosterman, M. Tamura, S. R. Holbrook, and S. E. Brenner, "SCOR: a Structural Classification of RNA database," *Nucleic Acids Research*, vol. 30, pp. 392–394, 01 2002.

[105] S. S. Ray, S. Halder, S. Kaypee, and D. Bhattacharyya, "Hd-rnas: An automated hierarchical database of rna structures," *Frontiers in Genetics*, vol. 3, p. 59, 2012.

[106] M. Andronescu, V. Bereg, H. Hoos, and A. Condon, "Rna strand: the rna secondary structure and statistical analysis database," *BMC bioinformatics*, vol. 9, p. 340, 09 2008.

[107] D. Kim, G. Pertea, C. Trapnell, H. Pimentel, R. Kelley, and S. Salzberg, "Tophat2: Accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions," *Genome biology*, vol. 14, p. R36, 04 2013.

[108] H. Yang, F. Jossinet, N. Leontis, L. Chen, J. Westbrook, H. Berman, and E. Westhof, "Tools for the automatic identification and classification of RNA base pairs," *Nucleic Acids Research*, vol. 31, pp. 3450–3460, 07 2003.

[109] I. K. K. L. H. K. A. G. E. A. B. P. J. K. G. C. R. D. F. A. B. A. K. S. G.-J. A. F. C. W. Z. B. Y. L. K. P. W. P. P. C. T. M. L. J. J. C. R. G. M. A. M. J. M. B. M. Y. N. K. B. C. J. R. C. M. S. W. M. K. V. W. E. H. T. Z. B. Y. Z. R. C. W. Z. M. D. P. I. S. V. A. G. H. L. M. Z. Z. J. P. P. F. S. D. M. S. B. P. F. S. R. E. J. M. C. P.-J. V. P. M. J. W. M. B. C. X. C. Q. M. E. D. The RNAcentral Consortium; Anton I Petrov, Simon J E Kay, "RNAcentral: a comprehensive database of non-coding RNA sequences," *Nucleic Acids Research*, vol. 45, pp. D128–D134, 10 2016.

[110] D. Dufour, E. Capriotti, and M. Marti-Renom, *Computational Methods for RNA Structure Prediction and Analysis*. 11 2013.

[111] M. Marti-Renom and E. Capriotti, "Computational rna structure prediction," *Current Bioinformatics*, vol. 3, pp. 32–45, 01 2008.

[112] G. Chojnowski, T. Walen, and J. M. Bujnicki, "Rna bricks - a database of rna 3d motifs and their interactions," *Nucleic Acids Research*, vol. 42, no. D1, pp. D123–D131, 2014.

[113] B. Carter, J. Fletcher, and R. Thompson, "Analysis of messenger rna expression by in situ hybridization using rna probes synthesized via in vitro transcription," *Methods (San Diego, Calif.)*, vol. 52, pp. 322–31, 12 2010.

[114] T. K. Mohanta, D. Yadav, A. Khan, A. Hashem, E. F. AbdAllah, and A. Al-Harrasi, "Analysis of genomic trna revealed presence of novel genomic features in cyanobacterial trna," *Saudi Journal of Biological Sciences*, vol. 27, no. 1, pp. 124 – 133, 2020.

[115] A. S. Walker, W. P. Russ, R. Ranganathan, and A. Schepartz, "Rna sectors and allosteric function within the ribosome," *Proceedings of the National Academy of Sciences*, vol. 117, no. 33, pp. 19879–19887, 2020.

[116] K. Lee and B.-J. Lee, "Structural and biochemical properties of novel self-cleaving ribozymes," *Molecules : A Journal of Synthetic Chemistry and Natural Product Chemistry*, vol. 22, 2017.

[117] R. Atilho, G. Mirihana Arachchilage, E. Greenlee, K. Knecht, and R. Breaker, "A bacterial riboswitch class for the thiamin precursor hmp-pp employs a terminator-embedded aptamer," *eLife*, vol. 8, 04 2019.

[118] C. Tin, D. Mai, P. Ngoc Diep, H. Le, and E. Lee, "Developments of riboswitches and toehold switches for molecular detection—biosensing and molecular diagnostics," *International Journal of Molecular Sciences*, vol. 21, p. 3192, 04 2020.

[119] G. Marshall, J. Feng, and D. Kuster, "Back to the future: Ribonuclease a," *Biopolymers*, vol. 90, pp. 259–77, 01 2008.

[120] S. Kanwar, P. Mishra, and R. Kumar, "Ribonucleases and their applications," *Journal of Advanced Biotechnology and Bioengineering*, vol. 4, 01 2016.

[121] D. Bechhofer and M. Deutscher, "Bacterial ribonucleases and their roles in rna metabolism," *Critical Reviews in Biochemistry and Molecular Biology*, vol. 54, pp. 242–300, 05 2019.

[122] N. Bradshaw and P. Walter, "The signal recognition particle (srp) rna links conformational changes in the srp to protein targeting," *Molecular biology of the cell*, vol. 18, pp. 2728–34, 08 2007.

[123] S. Srikamdee, W. Wattanapornprom, and P. Chongstitvatana, "Rna secondary structure prediction with coincidence algorithm," in *2016 16th International Symposium on Communications and Information Technologies (ISCIT)*, pp. 686–690, Sep. 2016.

[124] R. Nussinov and A. B. Jacobson, "Fast algorithm for predicting the secondary structure of single-stranded rna," *Proceedings of the National Academy of Sciences*, vol. 77, no. 11, pp. 6309–6313, 1980.

[125] M. Zuker and P. Stiegler, "Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information," *Nucleic Acids Research*, vol. 9, pp. 133–148, 01 1981.

[126] I. L. Hofacker, "Vienna RNA secondary structure server," *Nucleic Acids Research*, vol. 31, pp. 3429–3431, 07 2003.

[127] M. Zuker, "Mfold web server for nucleic acid folding and hybridization prediction," *Nucleic Acids Research*, vol. 31, pp. 3406–3415, 07 2003.

[128] H. Jabbari, I. Wark, and C. Montemagno, “Rna secondary structure prediction with pseudoknots: Contribution of algorithm versus energy model,” *PLOS ONE*, vol. 13, pp. 1–21, 04 2018.

[129] Y. Wu, B. Shi, X. Ding, T. Liu, X. Hu, K. Y. Yip, Z. R. Yang, D. H. Mathews, and Z. J. Lu, “Improved prediction of RNA secondary structure by integrating the free energy model with restraints derived from experimental probing data,” *Nucleic Acids Research*, vol. 43, pp. 7247–7259, 07 2015.

[130] K. Doshi, J. Cannone, C. Cobaugh, and R. Gutell, “Evaluation of the suitability of free-energy minimization using nearest-neighbor energy parameters for rna secondary structure prediction,” *BMC bioinformatics*, vol. 5, p. 105, 09 2004.

[131] L. Wang, Y. Liu, X. Zhong, H. Liu, C. Lu, C. Li, and H. Zhang, “Dmfold: A novel method to predict rna secondary structure with pseudoknots based on deep learning and improved base pair maximization principle,” *Frontiers in Genetics*, vol. 10, p. 143, 2019.

[132] X. Lu and W. K. Olson, “3DNA: a software package for the analysis, rebuilding and visualization of three-dimensional nucleic acid structures,” *Nucleic Acids Research*, vol. 31, pp. 5108–5121, 09 2003.

[133] F. Vendeix, A. Munoz, and P. Agris, “Free energy calculation of modified base-pair formation in explicit solvent: A predictive model,” *RNA (New York, N.Y.)*, vol. 15, pp. 2278–87, 10 2009.

[134] N. Nicolo, “Learning with kernels on graphs: Dag-based kernels, data streams and rna function prediction,” *Alma Mater Studiorum-Universitá di Bologna*, 2014.

[135] M. Zahran, C. Sevim Bayrak, S. Elmetwaly, and T. Schlick, “Rag-3d: a search tool for rna 3d substructures,” *Nucleic Acids Research*, vol. 43, pp. 9474–9488, 08 2015.

[136] N. Kim, M. Zahran, and T. Schlick, “Computational prediction of riboswitch tertiary structures including pseudoknots by ragtop: a hierarchical graph sampling approach,” in *Methods in enzymology*, vol. 553, pp. 115–135, Elsevier, 2015.

[137] C. Laing, S. Jung, N. Kim, S. Elmetwaly, M. Zahran, and T. Schlick, “Predicting helical topologies in rna junctions as tree graphs,” *PLOS ONE*, vol. 8, pp. 1–12, 08 2013.

[138] A. Petrov, C. Zirbel, and N. Leontis, “Automated classification of rna 3d motifs and the rna 3d motif atlas,” *RNA (New York, N.Y.)*, vol. 19, 08 2013.

[139] V. Reinharz, A. Soulé, E. Westhof, J. Waldispühl, and A. Denise, “Mining for recurrent long-range interactions in rna structures reveals embedded hierarchies in network families,” *Nucleic acids research*, vol. 46, 03 2018.

[140] C. G. Oliver, V. Mallet, P. Philippopoulos, W. Hamilton, and J. Waldispuhl, “Vernal: A tool for mining fuzzy network motifs in rna,” 09 2020.

[141] P. Kerpedjiev, C. Höner zu Siederdissen, and I. Hofacker, “Predicting rna 3d structure using a coarse-grain helix-centered model,” *RNA (New York, N.Y.)*, vol. 21, 04 2015.

[142] K. Darty, A. Denise, and Y. Ponty, “VARNA: Interactive drawing and editing of the RNA secondary structure,” *Bioinformatics*, vol. 25, pp. 1974–1975, 04 2009.

[143] Y. Ujiie, T. Kato, K. Sato, and Y. Matsuoka, “Curvature entropy for curved profile generation,” *Entropy*, vol. 14, no. 3, pp. 533–558, 2012.

[144] M. Kline, “Calculus: an intuitive and physical approach,” pp. 457–462, Courier Corporation, 1998.

[145] P. Verbeek and L. Van Vliet, “Curvature and bending energy in digitized 2d and 3d images,” in *8th Scandinavian Conference on Image Analysis, Tromso, Norway*, 1993.

[146] *Curvature (article).* (https://www.khanacademy.org/math/multivariable-calculus/multivariable-derivatives/differentiating-vector-valued-functions/a/curvature), Khan Academy, last accessed 14 April 2021.

[147] A. Mjaavatten, *Curvature of a 1D curve in a 2D or 3D space.* (https://www.mathworks.com/matlabcentral/fileexchange/69452-curvature-of-a-1d-curve-in-a-2d-or-3d-space), MATLAB Central File Exchange, Retrieved May 3, 2021.

[148] H.-S. Park and C.-H. Jun, “A simple and fast algorithm for k-medoids clustering,” *Expert Systems with Applications*, vol. 36, no. 2, Part 2, pp. 3336–3341, 2009.

[149] D. Nova and P. Estevez, “A review of learning vector quantization classifiers,” *Neural Computing and Applications*, vol. 25, 09 2014.

[150] P. Schneider, M. Biehl, and B. Hammer, “Distance learning in discriminative vector quantization,” *Neural computation*, vol. 21, no. 10, pp. 2942–2969, 2009.

[151] M. Neumann, R. Garnett, C. Bauckhage, and K. Kersting, “Propagation kernels: efficient graph kernels from propagated information,” *Machine Learning*, vol. 102, pp. 209–245, 2016.

[152] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, “Representation learning on graphs with jumping knowledge networks,” in *International Conference on Machine Learning*, pp. 5453–5462, PMLR, 2018.

[153] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, “Neural message passing for quantum chemistry,” in *International Conference on Machine Learning*, pp. 1263–1272, PMLR, 2017.

[154] S. Zhang, H. Tong, J. Xu, and R. Maciejewski, “Graph convolutional networks: a comprehensive review,” *Computational Social Networks*, vol. 6, 11 2019.

[155] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, “Struc2vec: Learning node representations from structural identity,” in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’17, (New York, NY, USA), pp. 385–394, ACM, 2017.

[156] H. Dai, E. B. Khalil, Y. Zhang, B. Dilkina, and L. Song, “Learning combinatorial optimization algorithms over graphs,” *arXiv preprint arXiv:1704.01665*, 2017.

[157] H. Gao and S. Ji, “Graph u-nets,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[158] H. Zhao, L. Jiang, J. Jia, P. Torr, and V. Koltun, “Point transformer,” *arXiv preprint arXiv:2012.09164*, 2020.

[159] B. Li, T. Zhang, and T. Xia, “Vehicle detection from 3d lidar using fully convolutional network,” *arXiv preprint arXiv:1608.07916*, 2016.

[160] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, “Multi-view 3d object detection network for autonomous driving,” pp. 6526–6534, 07 2017.

[161] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu, “Pct: Point cloud transformer,” *arXiv preprint arXiv:2012.09688*, 2020.

[162] N. Engel, V. Belagiannis, and K. Dietmayer, “Point transformer,” *arXiv preprint arXiv:2011.00931*, 2020.

[163] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, “3d shapenets: A deep representation for volumetric shapes,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1912–1920, 2015.

[164] Y. Li, S. Pirk, H. Su, C. R. Qi, and L. J. Guibas, “Fpnn: Field probing neural networks for 3d data,” *arXiv preprint arXiv:1605.06240*, 2016.

[165] D. Wang and I. Posner, “Voting for voting in online point cloud object detection,” 07 2015.

[166] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, “Volumetric and multi-view cnns for object classification on 3d data,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5648–5656, 2016.

[167] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, “Multi-view convolutional neural networks for 3d shape recognition,” in *Proceedings of the IEEE international conference on computer vision*, pp. 945–953, 2015.

[168] A. S. Ashour, Y. Guo, A. R. Hawas, and G. Xu, “Ensemble of subspace discriminant classifiers for schistosomal liver fibrosis staging in mice microscopic images,” *Health information science and systems*, vol. 6, no. 1, pp. 1–10, 2018.

[169] M. W. Lewis, A. Verma, and R. Hennig, “Predicting 3d rna folding patterns via quadratic binary optimization,” *arXiv preprint arXiv:2106.07527*, 2021.

[170] C. Nagao, N. Nagano, and K. Mizuguchi, “Prediction of detailed enzyme functions and identification of specificity determining residues by random forests,” *PLOS ONE*, vol. 9, pp. 1–12, 01 2014.

[171] R. Gao, M. Wang, J. Zhou, Y. Fu, M. Liang, D. Guo, and J. Nie, “Prediction of enzyme function based on three parallel deep cnn and amino acid mutation,” *International Journal of Molecular Sciences*, vol. 20, no. 11, 2019.