AN ABSTRACT OF THE DISSERTATION OF

Padideh Danaee for the degree of Doctor of Philosophy in Computer Science presented on December 6, 2019.

Title: Interpretable Machine Learning: Applications in Biology and Genomics

Abstract approved: ______________________________
David A. Hendrix

Machine learning (ML) and deep learning (DL) models impact our daily lives with applications in natural language modeling, image analysis, healthcare, genomics, and bioinformatics. The exponential growth of biological sequence data necessitates accompanying advances in computational methods. Although deep learning is highly effective for detecting and classifying biological sequences, challenges remain in extracting meaningful patterns and information from the learned models. To realize the potential of deep learning in biology, we need to develop strategies for model interpretation to reveal or further clarify biological principles. In this thesis, we first present problems and methods to classify patterns in biological sequence data. Next, we describe a series of techniques we developed to understand the machine learning models and identify meaningful biological patterns. For each problem we created an interpretable, intelligent system without sacrificing performance. To test our approaches for model interpretation, we first focused our analysis on known biological patterns, and then extended the search beyond what is known. This work can be categorized into four different applications: I) the development of bpRNA, a novel annotation tool capable of parsing RNA secondary structures. bpRNA is a richly-annotated database that contains over 100,000 structures from seven different sources along with base pairing information. II) The detection of pseudoknots from sequence data alone with a machine learning model, Pseudoknow. As one of the most common RNA structural motifs, pseudoknots are crucial for RNA regulation. Improving the prediction of RNA pseudoknot structure will allow for better understanding of how RNA structure informs regulation and metabolism. III) Classification from gene expression data using stacked denoising auto encoders (SDAE) to distinguish healthy cells from cancerous ones, and to predict post-mortem time-of-death. These classification methods were developed with the goal to identify genes that are most informative for prediction and hence most biological relevant. Our study suggests that the most influential genes from the dimensionality reduction performed by SDAE were highly predictive of cancerous vs non-cancerous cell type. IV) Interpretation of the rules learned by a deep convolutional neural network to recognize known and previously uncharacterized core promoter sequence motifs from the whole genome sequences of human. We proposed and compared new training strategies to identify transcription start sites (TSS), located within core promoters, from biological sequences. The main goal of this application was to develop new strategies to interpret how the convolutional neural network learns biological patterns, and to understand the correlations between and within the convolutional layers. These new techniques could aid in deriving unknown patterns in biology and genomics and are applicable more broadly to other areas of data science.

Interpretable Machine Learning: Applications in Biology and Genomics

by

Padideh Danaee

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented December 6, 2019
Commencement June 2020

Doctor of Philosophy dissertation of Padideh Danaee presented on December 6, 2019.

APPROVED:

Major Professor, representing Computer Science

Head of the School of Electrical Engineering and Computer Science

Dean of the Graduate School

Padideh Danaee, Author

ACKNOWLEDGEMENTS

I would like to begin my acknowledgments by quoting Albert Einsteins metaphor on science: The whole of science is nothing more than a refinement of everyday thinking. Almost five years have passed since I began pursuing my doctorate degree at Oregon State University. As time progressed I could see that graduate life, with all its challenges and difficulties, can be enjoyable and rewarding. I successfully completed my Ph.D. work; however, something has changed inside me. This experience has taught me how to come up with an idea, how to refine it with everyday thinking, and turn it into something meaningful for other scientists to utilize. I would like to take advantage of this opportunity to thank a few people to whom I believe I owe this achievement.

First, I would like to thank my dear supervisor, Dr. David Hendrix. David, over the past five years, you have taught me not only how to become a scientist, you have taught me how to stay focused, pay close attention to the details, enjoy discoveries, write a scientific manuscript, and turn it into a beautiful masterpiece. You also have taught me how to overcome hurdles by leading detailed technical discussions. I have always counted on your flawless advisory feedback and I will continue to learn from you.

My committee members, Dr. Prasad Tadepalli, Dr. Xiaoli Fern, Dr. Stephen Ramsey, Dr. Weng-Keen Wong, and Dr. Brett Tyler, who provided me invaluable feedback and support during the last couple of years. Special thanks to Dr. Christopher K. Mathews and Dr. Andy Karplus for their remarkable inputs to our manuscripts. My dear friend, Reza Ghaeini for all the enlightening and helpful discussions, suggestions, and collaboration. My dear friend and colleague, Lillian K Padgitt-Cobb who gave me insightful feedback on my thesis. My great friend, Rachael Kuintzle whose passion for science and new discoveries has motivated me to think differently and be persistent in research.

I would also like to thank all my great colleagues in the Hendrix lab for all the helpful discussions and great teamwork over the last few years: Lillian K Padgitt-Cobb, Rosalyn Fey, Dezhong Deng, Nathan Waugh, Junki Hong, and Jimmy Bell.

My special thanks go to my beloved husband, Navid Paydavosi, for his constant encouragement and my best friends, Golnaz Zandieh, Yasmin Ehteshami, Arghavan Salehian, and Mariam Okhovat, for all the emotional support and for cheering me to the finish line.

Last, but not least, I thank the Medical Research Foundation (MRF) and Oregon State University for financially supporting me during my graduate studies.

CONTRIBUTION OF AUTHORS

Mason Rouches, Michelle Wiley, Dezhong Deng, and Liang Huang contributed and reviewed the bpRNA manuscript.
Reza Ghaeini collaborated on deep cancer paper and provided insightful comments for the core promoter research.
Rosalyn Fey pre-processed the gene expression data and created the meta-data distribution plots in the human brain project. Junki Hong also contributed with the visualization plots.
Lillian K Padgitt-Cobb contributed to the core promoter project by editing the manuscript and initial wet lab experiments of yeast core promoters. Elise Van Fossen also helped with initial experiments on core promoters in Yeast.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

Figure Page

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF TABLES

LIST OF ALGORITHMS

Algorithm Page

LIST OF APPENDIX FIGURES

LIST OF APPENDIX FIGURES (Continued)

LIST OF APPENDIX FIGURES (Continued)

LIST OF APPENDIX TABLES

DEDICATION

Life is meaningless without a family…

To my beloved family; my parents, who are the source of support and encouragement, my husband *Navid*, who always stands by me and motivates me to pursue my dreams and overcome challenges, and my sisters *Shayesreh* and *Shiva* for their emotional support over the past five years.

Chapter 1: Introduction

1.1 Motivation

The exponentially increasing availability of genomics, transcriptomics, and other sequencing data opens tremendous opportunities for using statistical analysis, machine learning techniques, and deep learning models to solve biological problems. Machine learning (ML) and deep learning (DL) models are advantageous for analyzing large and complex biological data sets because they allow for identifying patterns and motifs at a genome-wide scale that could not be detected using other methods and without the bias of studying an isolated aspect of a complex biological process. Furthermore, the improvements are undeniable in domains such as natural language processing, image processing, healthcare, genomics, and bioinformatics. However, without understanding how decision-making occurs, the learned models are often dismissed as meaningless due to complexity and extended training time. Inspired by the work of others, we seek to reveal learned patterns from the black box and to understand how artificial intelligence (AI) model identified these patterns [79, 45]. These interpretable approaches can be broadly used for all data science problems.

The first and foremost step of any ML application is to understand the data through various exploratory data analysis (EDA) and visualization techniques. This primary step is necessary for utilizing the ML and DL models and is considered the first step toward model interpretability. Improved model performance can also be achieved when we understand the attributes of data, including volume, variables, missing information, and outliers, and then apply proper pre-processing steps. Furthermore, patterns observed in the data can potentially inform and improve the selection of features. Model interpretability is achievable as we are building the model. The explanation of results and the degree of each feature's contribution are delivered as part of the training. Rule-based ML models such as decision trees fall under this category, where the tree representation reveals the decision-making process. These types of models are interpretable by nature, and there is no need for additional steps to explain the results. This type of interpretability is common for traditional ML models that use pre-defined features as input.

It is essential to understand how DL models learn patterns because the network is a black box and does not require the selection of pre-defined features, in contrast with traditional ML approaches. Using a massive amount of data, DL models are trained to learn high-level features incrementally. High-performance DL models demand new techniques to identify how the network learns patterns and to understand how the black box identifies such patterns more deeply. Uncovering how complex neural networks identify patterns is also crucial to convey the value of these advanced techniques to the biology community. All the steps mentioned above are necessary to understand the problem and complex biological data better, to develop novel and improved solutions, and explain results and identify new patterns in biological sequences.

1.2 This Work

In this thesis, we present multiple approaches for how to incorporate an interpretable, high-performance ML, or DL model in genomics and bioinformatics, along with multiple applications. We show that understanding the AI black box not only opens the door to understanding biological patterns but also builds trust among biologists to take advantage of advances in computer science. Our results suggest that interpretable ML in genomics leads to a better understanding of how patterns in biological sequences inform structure and function.

1.3 Stages Of Work

The stages are as follows:

1 **bpRNA: large-scale automated annotation and analysis of RNA secondary structure.** While RNA secondary structure prediction from sequence data has made remarkable progress, there is a need for improved strategies for annotating the features of RNA secondary structures. Here we present bpRNA, a novel annotation tool capable of parsing RNA structures, including complex pseudoknot-containing RNAs, to yield an objective, precise, compact, unambiguous, easily-interpretable description of all loops, stems, and pseudoknots, along with the positions, sequence, and flanking base pairs of each such structural feature. We also introduce several new informative representations of RNA structure types to improve structure visualization and interpretation. We have further used bpRNA to generate a web-accessible meta-database, "$bpRNA - 1m$", of over 100,000 single-molecule, known secondary structures; this

is both more fully and accurately annotated and over 20-times larger than existing databases. We use a subset of the database with highly similar ($\geq 90\%$ identical) sequences filtered out to report on statistical trends in sequence, flanking base pairs, and length. Both the bpRNA method and the bpRNA-1m database will be valuable resources both for specific analysis of individual RNA molecules and large-scale analyses such as are useful for updating RNA energy parameters for computational thermodynamic predictions, improving machine learning models for structure prediction, and for benchmarking structure-prediction algorithms.

2 **Pseudoknow: a Method for Fast and Accurate RNA Pseudoknot Detection.** The functions of many RNAs are largely determined by their structures; hence, computational approaches to improve RNA structure prediction remains an active area of research. Many algorithms have been developed to predict well-nested RNA secondary structures (pseudoknot-free) with minimum free energy in polynomial time ($\mathcal{O}(n^3)$), and with overlapped pseudoknot structures in ($\mathcal{O}(n^4)$ and up to $\mathcal{O}(n^6)$). While many software applications have been developed to predict the secondary structure of RNAs, "PK-detection", the task of predicting the presence or absence of pseudoknots without regard to their location, has no known software solution. A machine learning solution to the PK-detection problem can predict whether a given RNA sequence is likely to possess pseudoknots in its secondary structure without computing it's structure. This approach can assist the choice of structure prediction software, thus leading to improvement in secondary structure prediction.

3 **A Deep Learning Approach for Cancer Detection And Relevant Gene Identification.** Cancer detection from gene expression data continues to pose a challenge due to the high dimensionality and complexity of these data. After decades of research there is still uncertainty in the clinical diagnosis of cancer and the identification of tumor-specific markers. Here we present a deep learning approach to cancer detection, and to the identification of genes critical for the diagnosis of breast cancer. First, we used Stacked Denoising Autoencoder (SDAE) to deeply extract functional features from high dimensional gene expression profiles. Next, we evaluated the performance of the extracted representation through supervised classification models to verify the usefulness of the new features in cancer detection. Lastly, we identified a set of highly interactive genes by analyzing the SDAE connectivity matrices. Our results and analysis illustrate that these highly interactive genes could be useful cancer

biomarkers for the detection of breast cancer that deserve further studies.

4 **Dimensionality Reduction of Gene Expression Data for Time-of-death Imputation.** There exist thousands of human gene expression post-mortem samples in publicly available data-sources without associated time-of-death (TOD). This critical roadblock prevents large-scale investigations on how daily rhythms of gene expression change with age and neurodegenerative diseases. This challenge creates a need for intelligent, systematic approaches to infer time of death, and hence, to enhance existing data sets and assist with revealing rhythmic patterns associated with aging and Alzheimer's disease. As an initial step towards accurate imputation of time of death, we developed a reliable method to classify gene expression human brain samples using denoising autoencoder model and support vector machine technique. We achieved an accuracy of 95.5% in the night/day binary classification task, which indicates the robustness of the denoising approach towards age, genotype, and environmental variations of the samples.

5 **Interpretation of the Rules Learned by a Deep Convolutional Neural Network to Recognize Core Promoters.**

Core promoters are genomic regions responsible for directing transcription initiation by RNA polymerase II machinery. Identifying core promoter elements is essential for understanding how transcription and gene expression is controlled at the DNA level. We developed a deep convolutional neural network (CNN) to precisely detect transcription start sites (TSSs) from DNA sequences. The performance of the model was tested against three different databases containing human core promoter sequences. Our deep CNN model achieved higher performance and generalization by incorporating K-shuffled adversarial negative samples to pre-train the convolutional filters. We also present how our new training strategy directed the network to learn complex and meaningful biological motifs in core promoter regions. More importantly, we propose a novel and systematic technique to identify meaningful biological patterns by studying the layers of the deep network. Our approach can be applied to other genomic contexts to answer and uncover insights about biology.

Chapter 2: Interpretability through Exploratory Data Analysis and Large-scale Automated Annotation of RNA Secondary Structures [1]

2.1 Introduction

Ribonucleic acid (RNA) is a type of macromolecule that is essential for all life, with functions including molecular scaffolding, gene regulation, and encoding proteins. The secondary structures and base-pairing interactions of RNAs reveal information about their functions [26, 36, 62, 110]. While RNA structure prediction has made tremendous improvements in the past decades, there are several limitations in available resources for researchers. While over 100,000 known RNA structures exist in various databases, the most detailed meta-database, RNA STRAND v2.0, contains less than 5,000 entries, and has not been updated in a decade. Moreover, even with base pairing data, the structural features present can be rather complex and there has not yet been a fully successful general approach presented to systematically resolve the structural topology and identify all structural features given the base pairing information. This limitation is part of the reason that most source databases do not provide dot-bracket sequences for their structures. Therefore, there is a need for reliable tools that identify and annotate structural features from RNA base pairing data.

We present "bpRNA" , a fast, easy-to-use program that parses base pair data into detailed structure "maps" providing relevant contextual data for stems, internal loops, bulges, multi-branched loops (multiloops), external loops, hairpin loops, and pseudoknots. Previous work to parse RNA structural topology from base pairs do not handle pseudoknots [150] or only analyze teritary structures [98]. bpRNA outputs new file formats (both high-level and detailed-level) for RNA secondary structures that provide information to help understand the structure. bpRNA has accurately generated the dot-bracket sequence for all structures, including the complex structures with pseudoknots.

The prediction of RNA secondary structure is based on thermodynamic model parameters that are calculated from available data of known structures [104, 10, 105]. Likewise, the study of RNA secondary structure creates a need for comprehensive meta-databases,

[1]A version of this chapter has been published [Danaee, Padideh, et al. Nucleic acids research, 2018]

the analysis of which could enable updated RNA thermodynamic parameters and prediction tools. The detailed structural annotations generated by bpRNA provide information needed to build a rich database of great use to the RNA research community. While databases of 3D structures exist[160, 170, 18], they don't serve the same needs as secondary structure databases. There have been many attempts at creating RNA secondary structure databases and meta-databases[112, 7, 112], all of the meta-databases except RNA STRAND v2.0[7] are no longer available or have not been updated. To meet this need, we have built a detailed meta-database, "bpRNA-1m" , consisting of over 102,318 single molecule (1m) RNA secondary structures extracted from seven different sources, and analyzed by bpRNA. These data, including the structure annotations provided by bpRNA, represent the largest and most detailed RNA secondary structure meta-database created to date and will be expanded as more data become available. This comprehensive meta-database can be used in machine learning applications, benchmarking studies, or can be filtered as desired for other RNA structure research.

2.2 Materials and Methods

2.2.1 RNA Secondary Structure Types

We use the term "stem" , as previously defined[7], to refer to a region of uninterrupted base pairs, with no intervening loops or bulges (Figure 2.1A). We label the two paired sequences of a stem as 5' or 3' depending on their order in the RNA sequence. A hairpin loop is an unpaired sequence with both ends meeting at the two strands of a stem region. The direction of the hairpin loop sequence also defines the nucleotides in the closing base pair and mismatch pair as being 5' or 3' (Figure 2.1B). An internal loop is defined as two unpaired strands flanked by closing base pairs on both sides, which are labeled as 5' vs 3' based on which is more 5' in the RNA sequence (Figure 2.1C). The closing base pair 5' of the 5' strand is labeled as the 5' closing pair, and the closing pair that is 3' of the 5' strand is the 3' pair. A bulge can be considered as a special case of the internal loop where one of the strands is of length zero (Figure 2.1D). Multi-branch loops (multiloops) consist of a cycle of more than two unpaired strands, connected by stems (Figure 2.1E). External loops are similar to multiloops, but are not connected in a cycle. Dangling ends are unpaired strands at the beginning and end of the RNA sequence.

Pseudoknots (PKs) are characterized by base-paired positions and (i', j') that satisfy

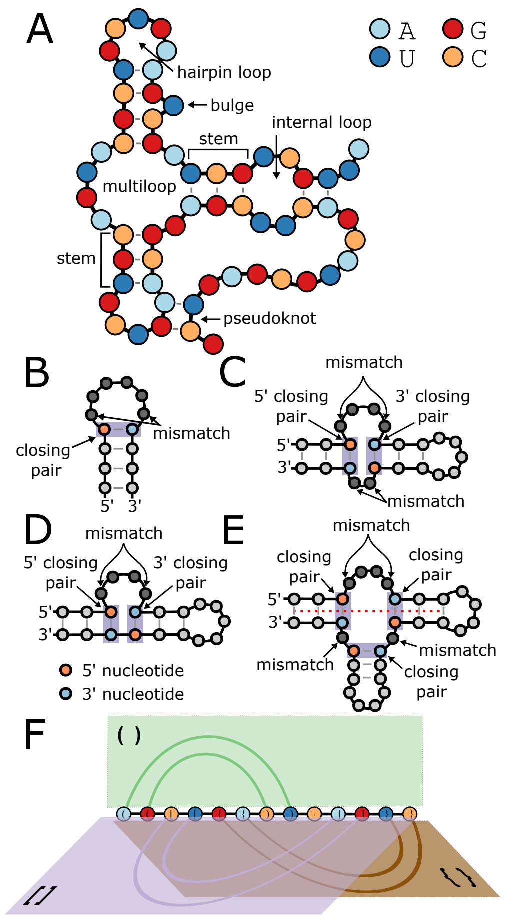


Figure 2.1: RNA Structure Types. **A.** cartoon schematic of RNA structure types. **B.** Hairpins have one closing pair and one mismatch pair with nucleotides defined by ordering from 5' to 3'. **C.** Internal loops have two closing base pairs and two mismatch pairs each defined by ordering from 5' to 3' relative to the 5' internal loop strand. The nucleotides of the closing pairs are defined as 5' or 3' based on their positions relative to the loop sequence. **D.** Bulges have one loop strand, but have two closing base pairs and two mismatch pairs defined 5' to 3'. **E.** Multiloops have a closing pair for each branch. The nucleotides of the closing pairs are defined as 5' or 3' based on their positions relative to the loop sequence. Red dashed line represents the common axis of coaxially stacked stems. **F.** A depiction of RNA page number, which can be viewed as separate half-planes containing edges corresponding to base pairs. Each symbol type corresponds to a separate page, and edges within a page are nested.

the PK-ordering, which is defined as either $i < i' < j < j'$ or $i' < i < j' < j$. For a secondary structure, PK base pairs are annotated as the minimal set that result in a PK-free structure when removed[150, 7, 180, 9, 165]. While representations of PK-containing RNA structures are not planar, the "book embedding" , or the number of distinct half-planes with a common boundary line (the RNA strand) can describe the RNA structure[63]. The number of half-planes needed to represent the structure is called the "page number" , and a book embedding for an RNA structure that has a lower page number is preferred because it provides a more compact representation. Figure 2.1F depicts the pages for an RNA structure with a page number of 3.

2.2.2 Segment Graph representation

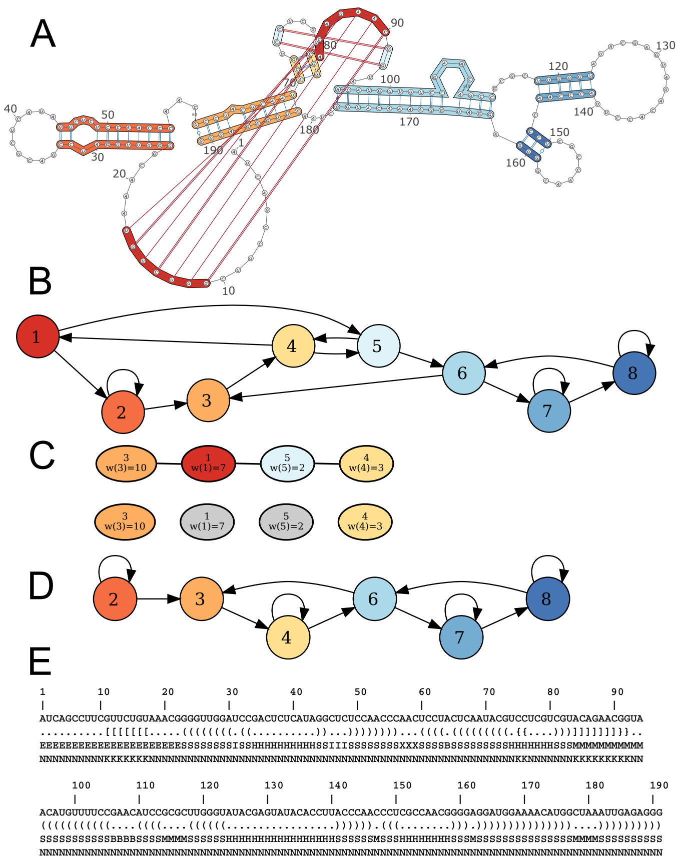
We have defined the "segment" and "segment graph" to assist in parsing RNA secondary structures and for visualization of structures. A segment is a region consisting of two strands of duplexed RNA that can contain bulges or internal loops. The difference between a stem and a segment is that segments can contain unpaired bases. When a base pair at positions (i, j) is part of a segment, then if the next paired nucleotide 5' of i is paired to the previous paired nucleotide 3' of j , then this base pair is also part of the segment. As an illustration of this idea, Figure 2.2A presents the structure of a ribozyme that contains 8 color-coded segments numbered 5' to 3'. This definition allows us to parse a structure into segments in linear time ("IdentifySegments" Algorithm 1 in appendix). The segment concept has some similarity to "bands" , which is loosely defined as "a pseudoknotted stem, which may contain internal loops or multi loops" [130], except segments apply more generally than pseudoknots, and do not contain multiloops. Pseudoknots (PKs) can be segments as well, such as segments 1 and 5 in Figure 2.2A-B, but the concept generally applies to any paired region.

The upshot of the segment representation is we can create a "segment graph" , which provides a compact representation of each structure (Figure 2.2B). Others have defined graph representations of RNA structures, such as "RNA As Graph" [52, 70]; however, the problem is these representations use stems as the edges of an undirected graph, making this extraordinarily complex for typical long noncoding RNAs, which can contain hundreds of stems or more. Moreover, examples from biology such as microRNAs show that many structures can preserve their functionality even when including bulges and internal loops[96]. These examples suggest a value in a more coarsely-defined secondary structure graph concept.

For any structure, we can define a directed multigraph $G = (V, E)$ such that the vertices V of the graph are the segments, the directed edges E correspond to unpaired strands, in the 5' to 3' direction, connecting them. Two segments can have an edge even when there is no intervening unpaired nucleotide (only a backbone). Each vertex of a segment graph can have at most two outgoing and two ingoing edges. Only the first and last segment can have less than two ingoing and outgoing edges. Hairpin stem-loops are easily identified as segments with self-edges, which count as one outgoing and one ingoing edge.

Pseudoknots (PKs) have been identified previously as the minimum set of base pairs that, when removed, produce a pseudoknot-free structure[150, 180, 9, 165], and algorithms have been developed for optimal selection of these base pairs[149]. We use the segment concept to identify this minimal set of base pairs. All pseudoknot base pairs are part of a segment, and these pseudoknot segments (PK-segments) can be easily identified; if one base pair of a segment satisfies the PK-ordering with a base pair in another segment, then all base pairs in this segment satisfy the PK-ordering with all base pairs in the other segment (See proof in appendix). Once PK-segments have been identified, a weighted, undirected graph called a PK-segment graph can be created such that the PK-segments correspond to vertices and edges connect them when they satisfy the PK-ordering with each other (Figure 2.2C). We assign a weight to each vertex, with the value of the number of base pairs for the PK-segment. From this graph, we next identify the maximum weighted independent subset (MWIS), leaving a minimal subset of segments whose removal leaves the secondary structure free of pseudoknots (Figure 2.2C). We created an exact algorithm, "MaxPKFreePairs" to selecting the MWIS using a Nussinov-style[118] dynamic programming approach similar to defined previously [150], as well as a heuristic algorithm "PK_Detection" for dealing with ties. We found that both methods produce the same solution to identifying the minimum subset of base pairs needed to produce a PK-free structure. These segments are then annotated as pseudoknots, and can be excised to produce a pseudoknot-free (PK-free) structure and PK-free segment graph (Figure 2.2D). The PK-free structure is equivalent to the page number=1 structure. The full algorithm for this approach is presented in Algorithms 2, 3, 4, and 5.

The PK-free segment graph enables facile identification of structure types. Hamiltonian cycles in the PK-free segment graph correspond to multiloops (Figure 2.2D). Interior loops and bulges can be identified as unpaired bases within segments. Pseudoknots are not discarded, but rather we annotate pseudoknots by the type of loops they connect in the



```
1        10        20        30        40        50        60        70        80        90
|        |         |         |         |         |         |         |         |         |
AUCAGCCUUCGUUCUGUAAACGGGGUUGGAUCCGACUCUCAUAGGCUCUCCAACCCAACUCCUACUCAAUACGUCCUCGUCGUACAGAACGGUA
..........[[[[[[.....((((((((.((..........))...))))))))...((((.(((((((((.{{....)))]]]]]]]}}..
EEEEEEEEEEEEEEEEEEEEEESSSSSSSSISSHHHHHHHHHHSSIIISSSSSSSSXXXSSSSBSSSSSSSSSHHHHHHHSSMMMMMMMMMMM
NNNNNNNNNNKKKKKKNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNKKNNNNNNNKKKKKKKKKNN

    100       110       120       130       140       150       160       170       180       190
     |         |         |         |         |         |         |         |         |         |
ACAUGUUUUCCGAACAUCCGCGCUUGGGUAUACGAGUAUACACCUUACCCAACCCUCGCCAACGGGGAGGAUGGAAAACAUGGCUAAAUUGAGAGGG
((((((((((....((((....((((((.................)))))).(((........))).))))))))))))))....))))))))))
SSSSSSSSSSSBBBBSSSSMMMMSSSSSSHHHHHHHHHHHHHHHHHHHSSSSSSMSSSHHHHHHHHHSSSMSSSSSSSSSSSSSSSSMMMMSSSSSSSSSS
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
```

Figure 2.2: Segment Graph example. **A.** Secondary structure of the *Anopholes gambia* drz-agam-2-2 ribozyme. **B.** The segments are the vertices of the segment graph and ordered from 5' to 3', and directed edges are defined by unpaired strands connecting segments. **C.** Segments with base pairs crossing other segments comprise the PK-graph. A maximally weighted independent set is selected by dynamic programming, with the remaining segments defined as pseudoknots. **D.** The pseudoknot-free segment graph is created after remove PK base pairs and allows easy annotation of loops. **E.** The structure array enhances bpRNA's multi-bracket dot-bracket sequence by labeling each positions structure type. Strands participating in pseudoknots are labeled in the structure array by their loop-type in the structure resulting from the removal of PKs.

corresponding PK-free structure. For instance, if a PK consists of base pairs connecting what would otherwise be a multiloop branch and a bulge, we label the PK as "M-B".

The bpRNA code is written in perl and requires the Graph perl module. Several additional scripts for analysis are included. The source code is available at http://github.com/hendrixlab/bpRNA.

2.2.3 Reference Databases

The seven databases that comprise the bpRNA-1m meta-database include Comparative RNA Web (CRW) [25], tmRNA database [190], tRNAdb [6], Signal Recognition Particle (SRP) database [88], RNase P database [23], tRNAdb 2009 database [74], and RCSB Protein Data Bank (PDB) [137], and all families from RFAM 12.2 [114]. Moreover, to reduce duplication for further analysis, we created a subset called bpRNA-1m(90), where we removed sequences with greater than 90% sequence similarity when there is at least 70% alignment coverage [94]. The bpRNA-1m database currently has 102,318 RNA structures and the bpRNA-1m(90) subset consists of 28,370 structures. For comparison, the RNA STRAND v2.0 database has 4,666 structures, with fewer than 2,000 structures when similarly filtered.

The Comparative RNA Web (CRW) site contains RNA sequences and secondary structures obtained from comparative sequence analysis. There are 55,600 records extracted from this reference through the mass data retrieval option. For each RNA extracted from this source, we retrieved phylogenetic lineage, organism name, and RNA type. The tmRNA Database provides structures of transfer messenger RNAs (tmRNAs), which are bacterial RNAs with both tRNA- and mRNA-like functions. The base pair information for all 728 RNAs from this source was also determined using comparative sequence analysis. Single Recognition Particle Database (SRP) is a source for structures and functions of single recognition particle RNAs (SRP RNAs) along with phylogenetic lineage and organism names for each RNA [6]. The tRNAdb 2009 database (formerly Sprinzl tRNA Database) encompasses all the structures and sequences from tRNA genes from three different university sources: Leipzig, Marburg, and Strasbourg [74]. All 623 of these verified RNA structures were downloaded from this source along with their taxonomy and links to each individual reference. The RNase P Database (RNP) has sequences and secondary structures of Ribonuclease P type RNA of bacteria, archaea, and eukaryotes. All available taxonomy, organism name, and associated PubMed ID data were downloaded for the 466 entries in this database.

RCSB Protein Data Bank (PDB) contains structures of proteins and nucleic acids obtained using X-ray crystallography and NMR techniques. We download all 669 RNA structures (PDB files) consisting of one RNA molecule as of June 12 2017. We first parsed the 3D structures from PDB files with the June 2017 version of RNAview [184], and used custom perl scripts to convert to BPSEQ format. This conversion considers both canonical and non-canonical base pairs. The priority is on the positions with Watson-Crick and Wobble pairs. The Watson-Crick pairs are identified by the edge represented in RNAView output (+/+ for GC pairs and -/- for AU pairs), and wobble pairs are recognized when the edge is Watson-Crick/Watson-Crick and has the *cis* orientation with XXVII Saengers classification [184]. Similarly, non-canonical pairs are extracted based on these three specifications [91].

The RNA Family Database (RFAM) V12.2 contains consensus structures derived from comparative sequence analysis of individual sequence family members of thousands of RNA families. For each sequence, we extracted the RNA type, validation technique and when available, the URL for the RNA family Wikipedia page. There are 2,495 families in RFAM V12.2 and 43,273 individual sequences. For each family, we projected consensus structures to individual sequences using multiple sequence alignments provide by RFAM and custom perl scripts. Base-paired positions in the consensus structure were mapped to individual sequences, while removing gaps in the alignment, as done in previous studies [7]. We include information on the publication status in the database for users that want to exclude unpublished structures.

The relational database is implemented with MySQL (Version 15.1) on a CentOS GNU/Linux server (Figure A.1). For more detail on the database, see Appendix for Methods.

2.3 Results and Discussion

2.3.1 The bpRNA approach

bpRNA Secondary Structure Decomposition and Representation. The input to bpRNA is a list of base pairs (BPSEQ file) for a given RNA secondary structure. First, the segments are identified as in Figure 2.2A,B. Next, a PK-graph is built, and the PK-segments are identified (Figure 2.2C). The PK-free segment graph, which enables multiloops and external loops to be easily identified, is built after the removal the base pairs in PK-segments (Figure 2.2D). Bulges and internal loops are identified as unpaired positions within the segments. After all loops are identified, the pseudoknots are annotated by the loops in the PK-free structure that they connect (See methods section in Appendix). The output of bpRNA analysis are 1) a multi-bracket dot-bracket representation of the secondary structure, 2) a "structure array" sequence providing more detail to the dot-bracket, and 3) a "structure type" file. The content of these files is described in the following sections.

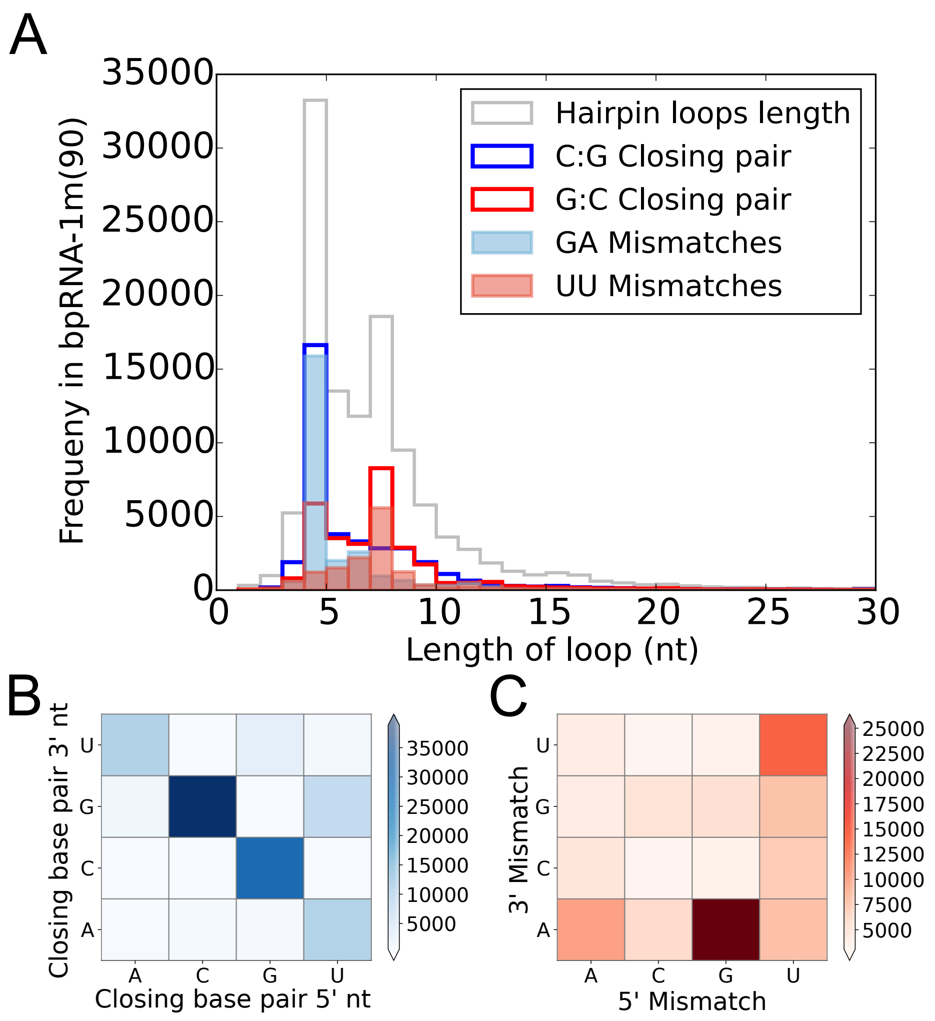


Figure 2.3: Hairpins in bpRNA-1m(90). **A.** The distribution of hairpin loop lengths in bpRNA-1m(90) has two primary peaks, overlapping the same peak for subsets defined by closing pairs. **B.** Heat map shows the frequency of nucleotides occurring in closing base pairs. **C.** Heat map shows the frequency of pairs of nucleotides occurring in hairpin mismatch pairs.

An Accurate Dot-bracket representation of RNA secondary structure. Dot-bracket format represents base pairs with paired parentheses, unpaired nucleotides with dots, and pseudoknots with other brackets ("[" ,"{ " ," <"...). While most of the databases that the data was derived from does not include a multi-bracketed dot-bracket representation when pseudoknots are present, bpRNA has successfully created one for every structure. Each dot-bracket representation we created is sufficient to re-create the BPSEQ file using our multi-stack approach to parse the dot-bracket structure. The efficiency of a dot-bracket sequence is described by the "page number" , which is the number of different symbol types used to represent the dot-bracket structure [34]. Our dot-bracket consist of dots "." for unpaired bases, matched parentheses indicate nested base pairs for page 1, square brackets for page 2, curly braces for page 3, angle brackets for page 4, and pairs of upper/lower alphabetical characters (Aa, Bb,. . . , Zz) for higher page numbers. Base pairs on the same page do not cross each other, i.e., each page is pseudoknot-free (Figure 2.1F). We were able to represent all structures with a page number less than or equal to 7, and 99.46% of the structures with a page number of 2 or less. For all 1,497 structures were bpRNA differs from RNA STRAND v2.0, bpRNA produced a lower page number lower page number, and thus a simpler dot-bracket sequence (Figure A.2). In some cases, RNA STRAND v2.0 had a page number as high as 30, requiring every letter of the alphabet to represent the pseudoknots of the structure, while bpRNA has a page number of 5.

The bpRNA "structure array" . We also created what we call the "structure array", which is a series of single character identifiers for the structure types of each nucleotide in the sequence, providing another layer of annotation to supplement the dot-bracket (Figure 2.2E) and a high-level representation of the structure. In this representation, S=stem, H=hairpin loop, M=multi-loop, I=internal loop, B=bulge, X=external loop, and E=end. The next sequence labels nested or unpaired nucleotides with "N" , and nucleotides forming pseudoknots with "K" . This enables a compact representation and additional detail, making the dot-bracket more easily interpretable for researchers. This is particularly helpful for loop regions, which are only represented as a dot "." in dot-bracket, and detailed by the type of loop with the structure array. Similarly, the structure array at pseudoknot positions indicates what loops result from the removal of the pseudoknots.

The bpRNA "structure type" file. We defined a new file format with each structural feature, relevant positions, and flanking base pairs, and sequences called a "structure type"

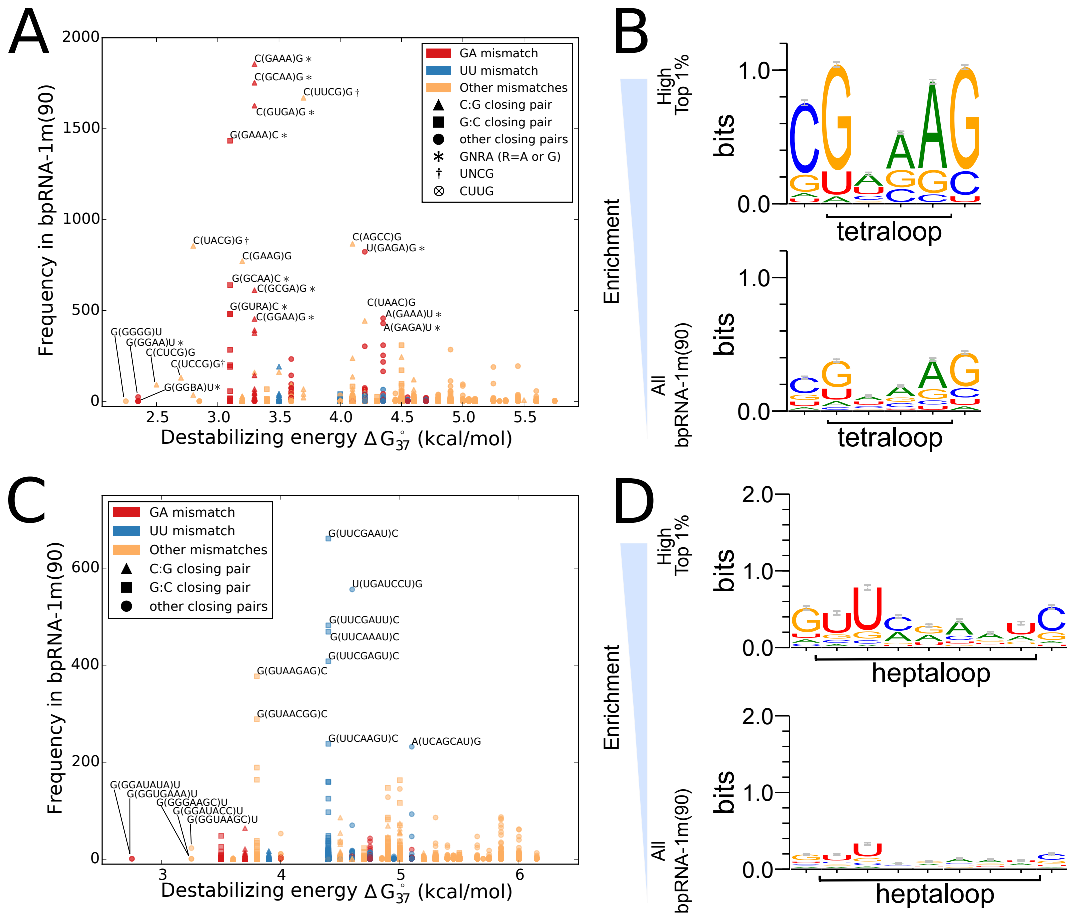


Figure 2.4: Tetraloops and Heptaloops. **A.** Scatterplot compares the frequency of tetraloop sequences to destabilizing energy. **B.** Sequence LOGOs demonstrate sequence biases in the most enriched tetraloops. **C.** Scatterplot compares the frequency of heptaloop sequences to destabilizing energy. **D.** Sequence LOGOs demonstrate the sequence biases in the most enriched heptaloops.

file (Figure A.3). This file format goes beyond the dot-bracket and structure array because it has more detail such as positions and flanking base pairs, and is capable of representing features of length zero, such as zero-length multiloop branches. Each feature is numbered, and PK interactions are indicated for loops that contain them. Researchers can unambiguously explore a structure with this information, along with the dot-bracket, structure array, and VARNA 2D structure image [41].

bpRNA yields accurately annotated features. We found a number of differences with our feature extraction and other work. For bulges, we found 1,042 structures with differences in the identified number of bulges. For instance, Figure A.4 shows a structure for the tmRNA *List.wels._AF440351_1-321*. This is annotated as having 0 bulges in RNA STRAND v2.0, but it actually has 4 bulges and these are all correctly annotated by bpRNA. In Figure A.5, we label in the structure diagram from the RNA STRAND v2.0 database with the bulges identified by bpRNA, and provide location and sequence information for these bulges. For other structure types, we have a different classification system. For example, if a hairpin loop participates in a pseudoknot (e.g. a "kissing hairpin pseudoknot"), RNA STRAND v2.0 does not annotate it as a hairpin. In contrast, we still classify loops by the above definitions even when they contain nucleotides forming PK base pairs, but label them with the specific PK involved. Furthermore, we categorize PK base pairs by the loop sequences that they connect.

Table 2.1: The number of RNAs from each source is listed for both bpRNA-1m and bpRNA-1m(90).

Data Source	**bpRNA-1m**	**bpRNA-1m(90)**
CRW	55,600	4,368
tmRNA	728	339
SRP	959	352
tRNAdb 2009	623	207
RNP	466	253
RFAMx	43,273	22,521
PDB	669	330
Total	102,318	28,370

2.3.2 The bpRNA-1m and bpRNA-1m(90) databases

The number of RNA structures extracted from each source is shown in Table 2.1 for both bpRNA-1m and bpRNA-1m(90). There are a relatively higher number of structures from CRW and RFAM database; however, around 92% of the CRW data are filtered running the CD-HIT-EST algorithm with the 90% similarity. In some cases, bpRNA detected errors in the source BPSEQ files used to build our meta-database: in cases where a nucleotide was paired to itself, the base pair was removed; in cases where a nucleotide was paired to two positions, one was removed. Overall, we found 30 such examples (Table A.1).

The complete bpRNA-1m database is available through our interactive website at `http://bpRNA.cgrb.oregonstate.edu`

2.3.3 Secondary Structure Feature Analysis

The output of bpRNA can help researchers understand RNA secondary structure, and enable large-scale structural analysis. As an example of the type of analysis that can be performed, we analyzed the resulting secondary structure annotations to identify enriched sequence and structural patterns in our database (Table 2.2). We performed this analysis on the bpRNA-1m(90) to reduce duplicated information. Table 2.3 shows the distribution of RNA types for bpRNA-1m and bpRNA-1m(90). We found several general trends in this large data set, which could be refined in future studies as more data become available, or with a more restricted subset.

Table 2.2: The number of each structures type for all RNA structures in bpRNA-1m and bpRNA-1m(90).

Structure Type	**bpRNA-1m**	**bpRNA-1m(90)**
Bulges	517,672	82,061
HairpinLoops	708,144	119,645
Multiloops	317,046	41,424
InternalLoops	538,670	93,435
Pseudoknots	57,686	7,164
ExteriorLoops	229,468	67,059
Stems	2,075,928	335,877
Segments	1,019,586	160,381

Hairpin Loops. The most common loop-type found in RNA secondary structures are hairpin loops [158]. For each hairpin loop, there is a closing base pair and unpaired region. The destabilizing energy of a hairpin loop can be determined from the type of the closing base pair, type of mismatch, and the length of the unpaired region [105, 145]. Using the bpRNA-1m(90), we found that tetraloops, hairpin loops of length four, are the most common (Figure 2.3A). While many previous studies focused hexaloops, loops of length 6 [105, 144, 67, 48], we found that heptaloops, hairpin loops of length seven, are the second most frequent (Figure 2.3A). Hairpin loops of size less than 4 and greater than 7 occur much less frequently in bpRNA-1m(90).

Table 2.3: The number of common RNA types is listed for bpRNA-1m and bpRNA-1m(90).

RNA Type	**bpRNA-1m**	**bpRNA-1m(90)**
$TransferRNA$	35,622	3,383
$16SRibosomalRNA$	17,641	1,067
$5SRibosomalRNA$	477	607
$SignalRecognitionParticleRNA$	1,603	388
$RibonucleasePRNA$	1,425	605
$TransferMessengerRNA$	161	449
$GroupIIntron$	237	123
$23SRibosomalRNA$	191	72
$HammerheadRibozyme$	186	77
$GroupIIIntron$	131	101

When considering all hairpin loops in bpRNA-1m(90), we found that C:G followed by G:C are the most common closing base pairs (Figure 2.3B), and GA mismatches are the most common overall (Figure 2.3C). The data suggest that tetraloops are significantly enriched with C:G closing base pairs, while heptaloops are enriched with G:C closing pairs. The tetraloops with C:G base pairs are mostly associated with GA mismatches, while heptaloops of size seven have the G:C base pair followed by UU mismatch. There are known frequent and stable patterns for tetraloops from various studies such as UNCG, GNRA, and CUUG, where N=A, C, G, or U and R=A or G [144, 133, 164]. Previous work has compared the statistical frequency of secondary structural features to thermodynamic stability [179, 54]. Using Turner 2004 nearest neighbor model, we compared the destabilizing energy of the hairpin loop types to their frequency of occurrence in bpRNA-1m(90) (Figure 2.4A). As it is shown, the GNRA and UNCG patterns are highly abundant whereas CUUG was not as frequent in our set. Sequence LOGOs [39] for all tetraloop tokens and for the top 1% when sorted by type frequency is presented in Figure 2.4B. We also did the same energy calculation for heptaloops, which is illustrated in Figure 2.4C along with sequence LOGOs in Figure 2.4D. Altogether in bpRNA-1m(90), the most common type for tetraloops is C(GAAA)G and for heptaloops the most common type is G(UUCGAAU)C (Figure 2.4). In other examples, there are loops that have low energy and a low frequency of occurrence. For example, G(GGUAAGC)U is probably rare because it is more stable

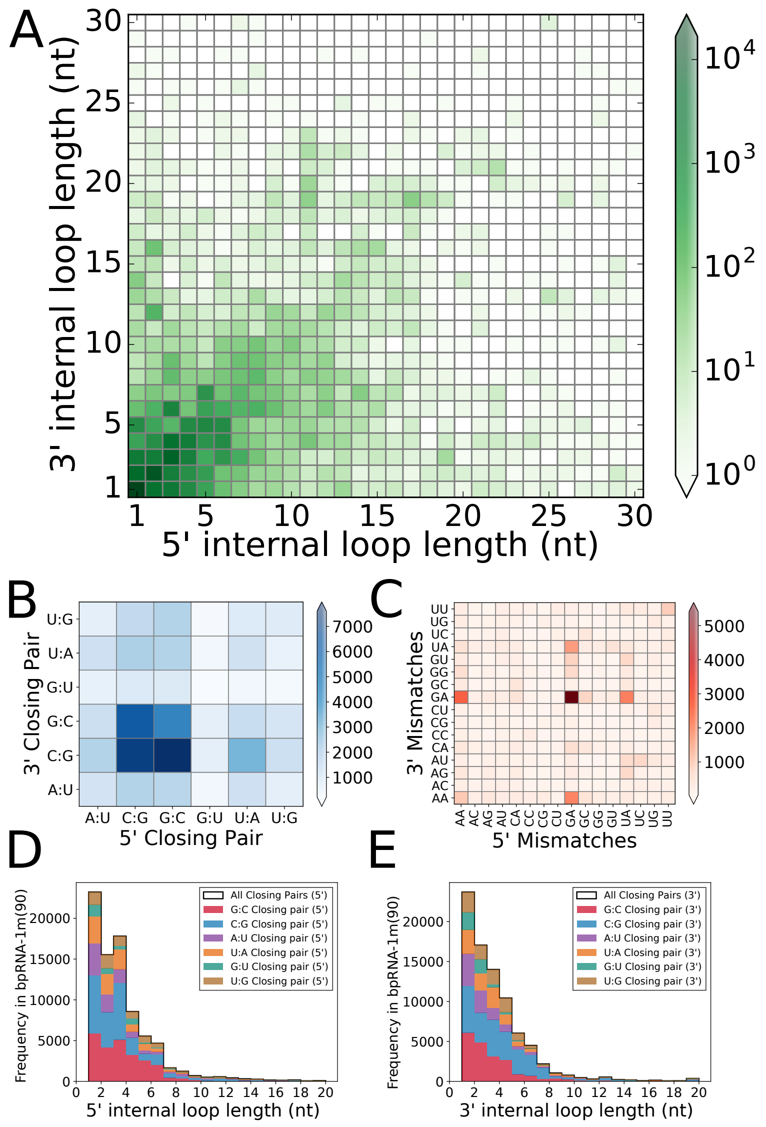


Figure 2.5: Internal loops. **A.** Heat map shows the frequency of internal loops based on 5' and 3' loop length. **B.** Heat map shows the frequency of base pairs occurring in 5' and 3' internal loop closing base pairs. **C.** Heat map shows the frequency of pairs of nucleotides occurring in 5' and 3' internal loop mismatch pairs. **D.** Stacked histograms of 5' internal loop lengths when organized by the 5' closing base pair. **E.** Stacked histograms of the 3' internal loop lengths when organized by the 3' closing base pair.

for the GC mismatch to pair, forming a loop of length 5.

Internal Loops. Internal loops tend to be symmetric, because this creates a more stable structure [123]). The internal loop frequency heat map (Figure 2.5A) demonstrates a tendency toward symmetric internal loops in bpRNA-1m(90), particularly when fewer than 4 nt. There are various factors in calculating the energy parameters of an internal loop such as first mismatch, closing base pairs, and the length of the 5' and 3' loop sequences [105]. We found that while the 5' closing base pair favors G:C, the 3' closing base pair favors C:G (Figure 2.5B). Mismatch nucleotides, defined as the first and last nucleotide of the loop, are enriched for GA (Figure 2.5C). Moreover, we found that internal loops with GA mismatches were most likely to have a length of 3 (Figure A.6). We found that 5' and 3' internal loops had slightly different length distributions, with 5' showing a greater propensity for length 3 (Figure 2.5D,E). The preference for C:G for the 3' closing pair is especially true for 3' internal loops longer than 3 nt (Figure 5E).

Bulges. The bulge length distribution obeys an approximate exponential distribution (Figure 2.6A) consistent with the destabilizing energy of a bulge increasing as a function of length. When the bulge loop is of length 1 nt, the nucleotide is enriched for A, and depleted for G and C, when compared to global nucleotide frequencies in this database (Figure 2.6B). The strongest deviation from the exponential fit is at length 6 nt, which is also enriched for bulges with a GA mismatch (Figure A.7A).

Similar to internal loops, bulges show the highest enrichment for G:C at the 5' and C:G at the 3' closing pairs (Figure 2.6C). The majority of bulges are flanked by GC base pairs, but for bulge loops less than 3 nt other flanking base pairs are common (Figure A.7 B-D). In addition, while GA mismatches are the most common for internal loops, the most common mismatch for bulges is AA, with GA the second most common (Figure 2.6D). The largest asymmetry between 5' and 3' closing base pairs was observed for U:G 5' closing pairs for bulges less than 4 nt (Figure 2.6E,F). Internal loops and bulges show similar trends for lengths when binned by closing pairs, but with bulges having a more sharply decaying distribution.

Multiloops. Based on analysis of bpRNA-1m(90), we found that multiloops branches (junctions) of size 3, 4, and 5 are the most common and multiloops of greater than size 6 nt are very rare (Figure 2.7A). Additionally, the distributions of branch lengths for these common multiloop branch-counts indicates that multiloops with 4 branches are significantly enriched for multiloop branches of zero length (Figure 2.7B), which is found in "flush stacking" [165]. This pattern is consistent with the fact that multiloops with four branches have more opportunities to be stabilized by coaxial stacking when the branches are zero length. In contrast, two helices in a multiloop with three zero-length branches would still be offset asymmetrically by the width of the third helix.

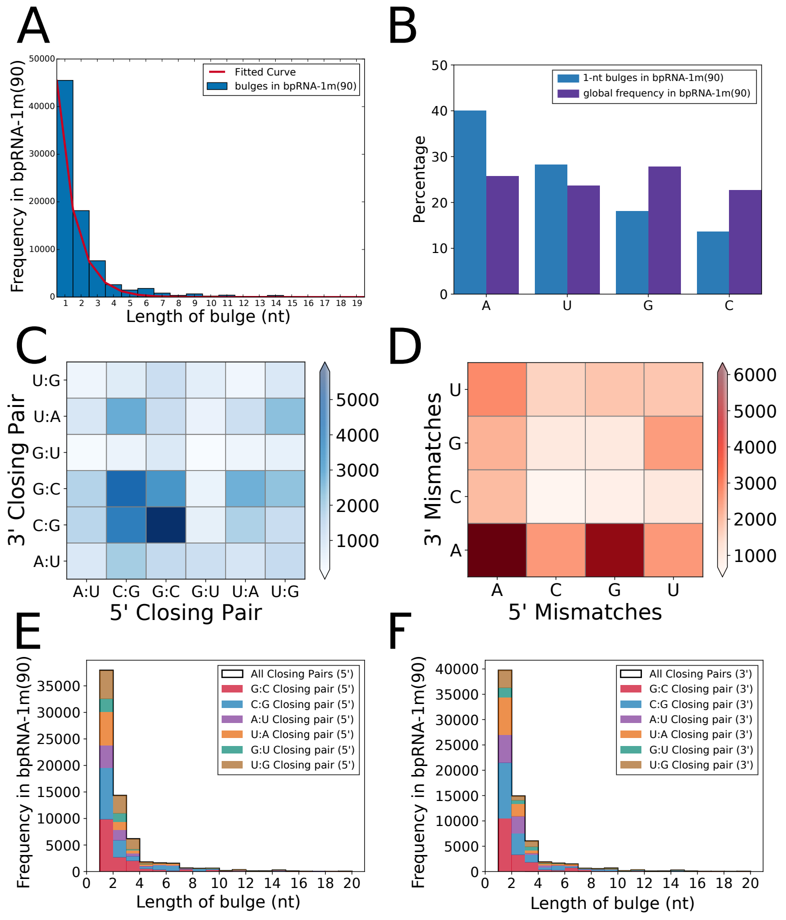


Figure 2.6: Bulges. **A.** Bulge length histogram. **B.** Nucleotide frequency in bulges of length 1. **C.** Heat map of closing base pairs. **D.** Heat map of mismatches. **E.** Bulge length distribution for different 5' closing base pairs. **F.** Bulge length distribution for different 3' closing base pairs.

Heat maps of the frequency of each closing base pair in multiloops branches demonstrates that most of the closing base pairs in multiloops tend to be C:G for 5' closing pairs, and G:C for 3' closing pairs—the opposite of internal loops (Figure 2.7C). This pattern for the closing pairs is the most common regardless of the number of branches (Figure A.8A-C). Overall, G:C and C:G closing pairs are significantly more common (Figure A.8D-I). In contrast to both internal loops and bulges, the most common mismatch pair for multiloops is AG (Figure 2.7D). Multiloop branches have a strong preference for GC-base pairing, with loops of length 0 showing a preference for C:G closing pairs, and loops of length 2 showing a preference for G:C closing pairs (Figure 2.7E-F).

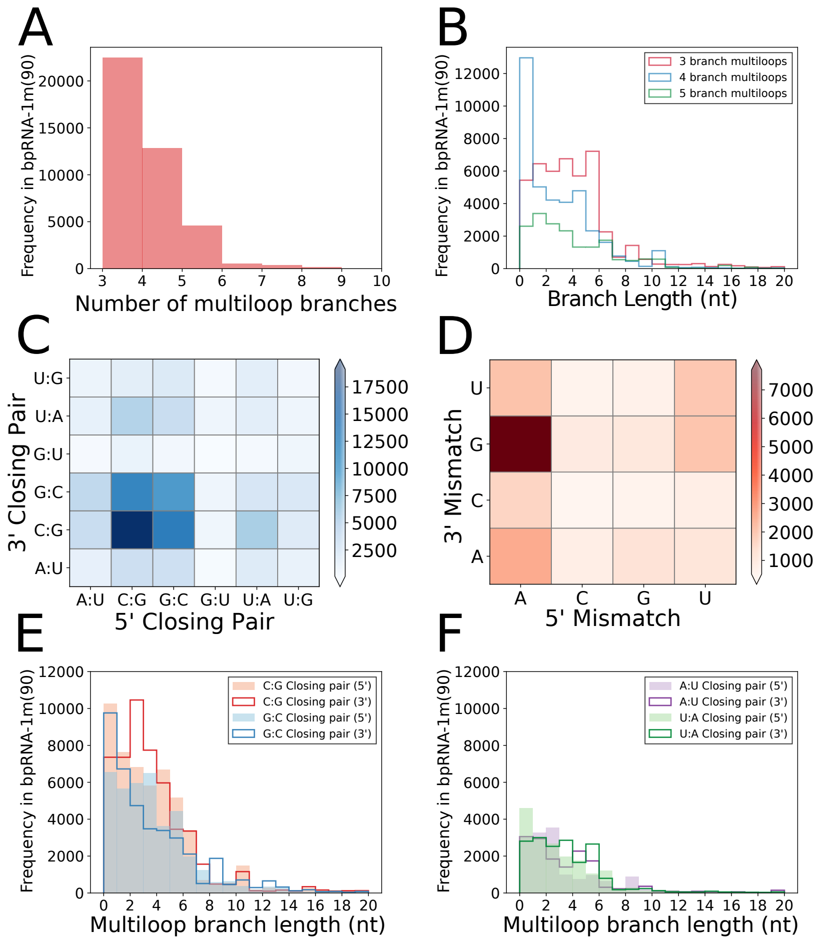


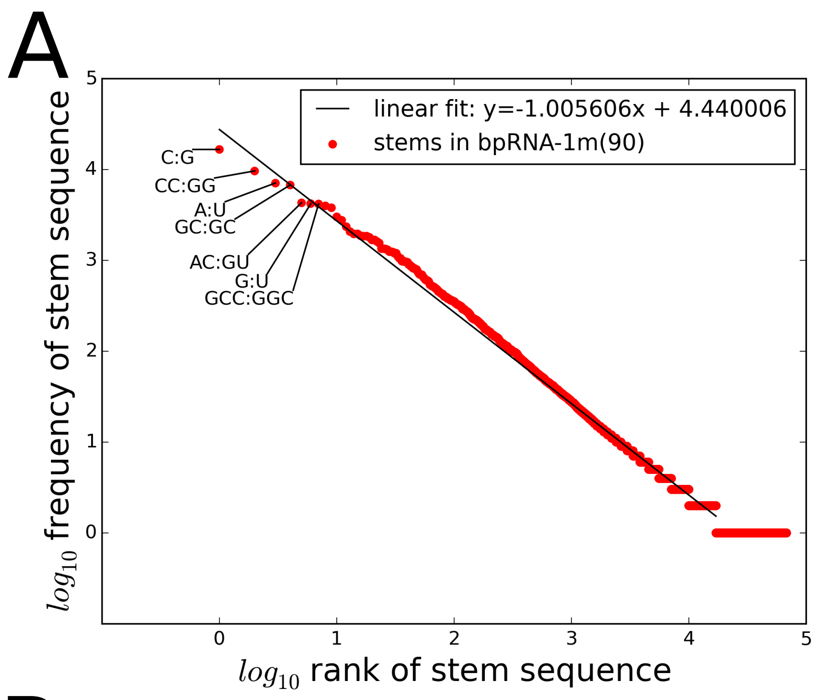
Figure 2.7: Multiloops. **A.** Histogram of branch number for all multiloops in bpRNA-1m(90). **B.** Branch length for multiloops with different branch numbers. **C.** Closing pair heat map. **D.** Mismatch heat map. **E.** Length distribution for different GC closing base pairs **F.** Length distribution for different AU closing base pairs.

Stems. Each stem in the database can be considered an instance of a “stem type” , such as CAG:CUG. To avoid double-counting, we alphabetically sort the two strands to form a distinct type. The full bpRNA-1m database contains of 2,075,928 stems that are instances (tokens) of 44,307 stem types, and bpRNA-1m(90) has 335,877 stems and

34,424 stem types. The frequency of stem type occurrences obeys a Zipfian distribution [189, 124], as observed in Figure 2.8A. The frequency f of occurrence of stems follows the equation, f=Ar^{-s} , where r is the rank of the stem when sorted by frequency, and the scale factor $s \approx 1.005$, extremely close to the idea Zipf relationship of $s = 1$. The frequency of occurrence of stems does not correlate perfectly with the energy of the stem sequence, because longer stems are typically less frequent.

Pseudoknots. Around 12% (3,320) of RNA structures have at least one pseudoknot (PK) in bpRNA-1m(90) (Table A.2). Most PK-containing RNAs have only one PK; however, many RNA secondary structures contain more than one PK. Overall, there are 7,164 PKs in this data set. To get a sense of most frequent loop types forming the PK structures in our set, we plotted the frequency of each type of PK in Figure 2.8B. The most frequent type is between multiloops and hairpin loops, followed by bulges and hairpin loops. kissing hairpins (H-H), which are commonly studied [154, 27], are the 7th most common. Consistent with our expectations that dangling ends and external loops cannot form pseudoknots with each other because such an interaction would form a multiloop and not a PK, our annotations do not find any examples of this. Analyzing base pair information per pseudoknot structures suggests that PKs with three base pairs are the most frequent in our dataset and there are only four PKs in bpRNA-1m that have 12 base pairs, the largest observed in bpRNA-1m (only one PK with length 12 observed in bpRNA-1m(90)).

Non-Canonical Base Pairs. The C:G/G:C base pairs in both bpRNA-1m and bpRNA-1m(90) outnumber any other base pairs. In addition to Watson-Crick (base pair interaction between C and G or A and U) and wobble base pairs (G:U pairs), there are other nucleotide interactions observed in the databases we have compiled, commonly referred to as non-canonical base pairs. Even though the canonical base pairs (Watson-Crick and wobble pairs) are more common in RNA secondary structures formation, non-canonical base pairing is important in the formation of the tertiary structures. We observe 9.1% of the base pairs in bpRNA-1m(90) are non-canonical. In 44.8% of these non-canonical pairs occur in the middle of a stem surrounding by canonical pairings, whereas only 7.2% are isolated base pairs. Also, about 1.4% of these special pairings are involved in pseudoknot formation (All stats are based on bpRNA-1m(90)). Table 2.4 shows the frequency of each type of base pairs in bpRNA-1m and bpRNA-1m(90). A:G/G:A, and A:C/C:A are the most common non-canonical pairs in both bpRNA-1m and bpRNA-1m(90), and C:C are the least frequent.



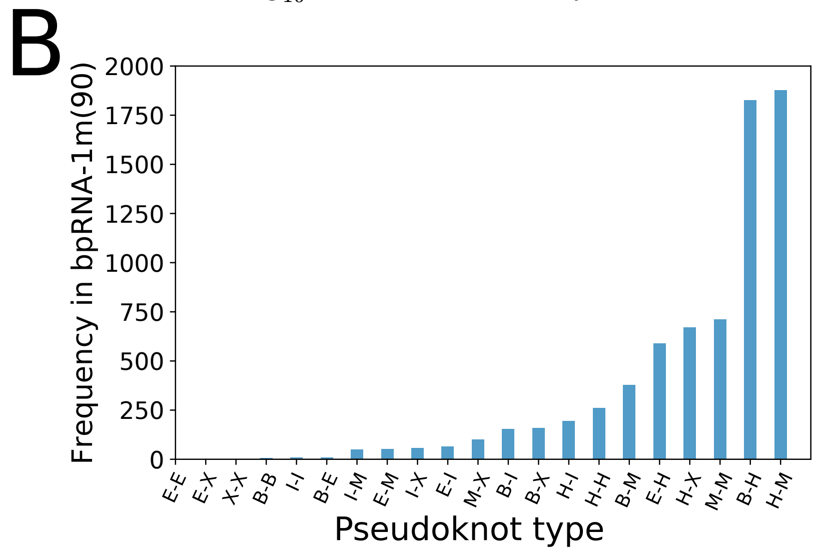


Figure 2.8: Stems and Pseudoknots. **A.** The frequency of stem types compared to their rank has a Zipfian distribution with a scale factor approximately equal to -1.00. **B.** bpRNA classifies pseudoknots by the loops that their base pairs connect when the pseudoknots are removed.

Table 2.4: Number of canonical and non-canonical base pairs in bpRNA-1m and bpRNA-1m(90).

RNA Type	**bpRNA-1m**	**bpRNA-1m(90)**
$C : G$	5,027,894	747,110
$A : U$	2,232,052	410,641
$G : U$	1,137,821	174,545
$A : G$	239,066	32,564
$A : C$	105,964	26,074
$U : U$	87,396	18,958
$C : U$	56,063	18,587
$G : G$	51,959	11,820
$A : A$	39,072	10,499
$C : C$	21,421	7,748

2.4 Summary and Conclusions

We have developed the bpRNA annotation approach to reliably produce intuitive secondary structure annotations from base pairing data to help with understanding RNA structure. Our efforts to provide annotations that are more informative and generally applicable than previous approaches have yielded many new strategies for representing RNA structural data such as the structure array, which makes the structure easier to read and visualize by providing a character label for each nucleotide of the dot-bracket representation. Likewise, the structure type file represents a detailed annotation, covering each nucleotide of the sequence. Separating the structure into segments—base paired regions interrupted by only bulges and internal loops—provides facile identification of multiloops and external loops, even when their length is zero. bpRNA also creates accurate dot-bracket representations for both simple and complex pseudoknot-containing RNA secondary structures.

We applied bpRNA to create a large integrated meta-database of single molecule RNA secondary structure that we have assembled from seven different sources (bpRNA-1m). With this large meta-database and the RNA structural information that bpRNA provides, there is an opportunity for a number of applications. The annotations produced from bpRNA could be used to improve the source databases used to build bpRNA-1m. Expanded structure annotations could enable the calculation of a next generation of thermodynamic parameters. The data set generated by bpRNA is large enough to enable training and testing machine learning algorithms for the prediction of RNA structure. Moreover, by restricting to only include single molecule structures, this dataset can serve as a benchmark for RNA secondary structure prediction algorithms, which typically take a single sequence as input.

We have used the annotation details and structural features produced by bpRNA to identify several statistics trends in bpRNA-1m(90), which contains over 28,000 sequences that are less than 90% similar, over 10 times the size of previous similar refined data [126]. While some of these trends represent patterns of thermodynamic stability, future studies are needed to expand this analysis with more structures, or judiciously filter the data for a more refined structural analysis.

DATA AVAILABILITY All data and scripts are accessible to download in `http://bprna.cgrb.oregonstate.edu/index.html`

Chapter 3: Model Interpretability as Part of a Rule-based method for Fast and Accurate RNA Pseudoknot Detection

3.1 Introduction

Ribonucleic acids (RNAs) are highly abundant molecules in living systems, with functions ranging from gene regulation, recruitment of molecular complexes, scaffolding, as well as encoding proteins [175]. Recent evidence suggests that genes for noncoding RNAs, which act functionally as transcripts and do not encode proteins, outnumber protein-coding genes in the human transcriptome [69]. The increasing abundance of noncoding RNA annotations highlights the need for fast and accurate approaches for genome-wide analysis of RNA structure. RNA secondary structure is an important first approximation of the tertiary structure and can represent a great deal of the RNA functionality. Secondary structures are characterized by the hydrogen bonds formed between base pairs. The common possible base pairings are Watson-Crick, (A, U) and (G, C), and Wobble (G, U). These structures can be divided into two broad classes: nested structures containing stems and loops, called pseudoknot-free (PKF) (Fig 3.1A), and complex overlapped structures, called pseudoknots (PKs) (For example, Fig 3.1B depicts one of the simplest classes of pseudoknots, called H-type [175]). Hereafter, we will only be concerned with secondary structure, and therefore for simplicity we will use the term "structure" when referring to "secondary structure".

The general problem of RNA structure prediction (RNA folding) with pseudoknots is NP-hard [3, 43], and several algorithms have been developed to address this challenge. The complexity of the current best practices for PK-structure prediction ranges from $\mathcal{O}(n^4)$ to $\mathcal{O}(n^6)$ in time and from $\mathcal{O}(n^2)$ to $\mathcal{O}(n^4)$ in space, which can be formidable, especially for longer sequences. Alternatively, no software approaches exist for "PK-detection", the challenge of predicting the presence or absence of PK-structures without actually computing their structures.

We present a solution to the PK-detection problem called `Pseudoknow`, which uses features extracted from RNA sequences in $\mathcal{O}(n)$ time, and accurately differentiates RNAs containing pseudoknots from those that do not, utilizing Machine Learning (ML). By distinguishing PK-structures from PKF-structures, one can choose an appropriate tool to predict RNA structures and improve the overall accuracy of prediction. The benefit of this approach is especially significant in genome- and transcriptome-wide studies where the computational prediction of RNA structure is costly, as it allows transcripts to be placed into PK and PKF bins to be further analyzed, thereby preventing unneeded computational complexity and improving structure prediction by selecting the appropriate algorithm. In addition, the analysis of the feature importance of this approach could provide further insight into PK formation useful in RNA folding.

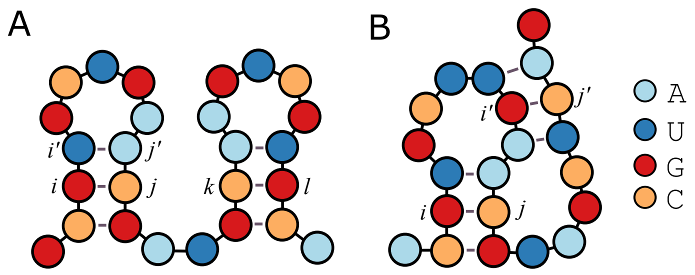


Figure 3.1: RNA secondary structures representation. **A.** A secondary structure of an RNA without a pseudoknot that has base pair (i, j) with all other base pairs such that $i < i' < j' < j$ as with (i', j'), or $i < j < k < l$ as with (k, l). **B.** A secondary structure with a pseudoknot and base pairs (i, j) and (i', j') such that $i < i' < j < j'$.

One can predict restricted classes of RNA structures without overlapped base pairings in polynomial time with the `mfold` and `ViennaRNA` software packages ($\mathcal{O}(n^3)$) [65, 106]. Although these programs are very powerful in predicting some classes of RNA structures, they do not predict pseudoknot base pairs. However, in many cases it is important to compute PK-structures since they appear to be in various types of RNAs such as transfer-messenger RNAs (tmRNA), ribosomal RNA (rRNA), and viral RNAs [169]. Pseudoknot structures are known to be involved in the formation of ribozymes, self splicing introns, telomerase RNAs, riboswitches, and can contribute to the process of ribosomal frameshifting mainly in viruses [131, 163, 1, 111, 147, 116, 156]. There exist multiple algorithms that predict different types of PK-structures and, depending on the type of the PK, the running time of these algorithms can vary from $O(n^4)$ to $O(n^6)$ [136, 101, 3, 166].

However, for a newly discovered RNA sequence, where nothing is known about the structure, it is not obvious which algorithm would give the most accurate structure prediction. If, for example, the sequence has a PK-structure and we run software that does not predict pseudoknots, the predicted structures do not tell us whether the sequence has a PK potential or not. Using a program that predicts PK-structures on an RNA that actually produces a PKF structure could use unnecessary resources, and in many cases predict erroneous PK-structures. It is the lack of available software for classifying RNAs by their potential to form PK-structures that motivated us to create `Pseudoknow`.

3.2 Materials and Methods

3.2.1 RNA Structure Data

In order to train and test our ML models, we used the sequences from CompaRNA benchmarks [127]. CompaRNA provides an established benchmark data set for RNA structural studies, and contains abundant PK- and PKF-structures. The CompaRNA data are derived from the RNA STRAND v2.0 database, which compiles various types of experimentally derived RNA structures from 6 different sources [8]. The CompaRNA set is produced by removing sequences less than 20 nucleotides and redundant sequences from RNA STRAND v2.0. These redundant sequences are filtered using `CD-HIT-EST` [95] by requiring less than 90% sequence identity, and a value of 70% for the minimal alignment coverage of the longer sequence. The data set contains 1987 sequences, 1068 of which are PKF structures and 919 are PK.

Because the RNA folding methods examined here are designed for predicting structures of a single RNA molecule, we further filtered CompaRNA to exclude structures that contain multiple nucleic acid sequences base-paired to each other. For example, PDB_00018 and PDB_01194 are “multi-molecular structures” that have annotated secondary structures (bpseq files) in RNA STRAND v2.0/CompaRNA that depict them as single molecules. Furthermore, these two accessions and many others are annotated as having pseudoknots, even though the base pairs leading to this designation are intermolecular, hence technically not pseudoknots. This filter results in 1760 sequences ranging from 21 to 4000 nt long, among which 865 are PK- and 895 are PKF-structures. We refer to this data set as “CompaRNA-1m” (1m = one molecule) to differentiate it from the full CompaRNA data set. We did most of our analysis on CompaRNA-1m; however, the results from the original CompaRNA data set also summarized in B.1 and B.2.

3.2.2 Feature Extraction

Central to our approach is the extraction of features to be used as input for our ML models. We have defined 103 features that can be extracted from the RNA sequence in linear time. Consider an RNA sequence x of length $|x| = n$, where individual residues are identified by $x[i] \in \{A, C, G, U\}$ and subsequences by $x[i..j]$.

We begin with some basic composition-based features, including GC-content, 16 features for di-nucleotide composition (dimer frequencies), and 64 features for tri-nucleotides (trimer frequencies).

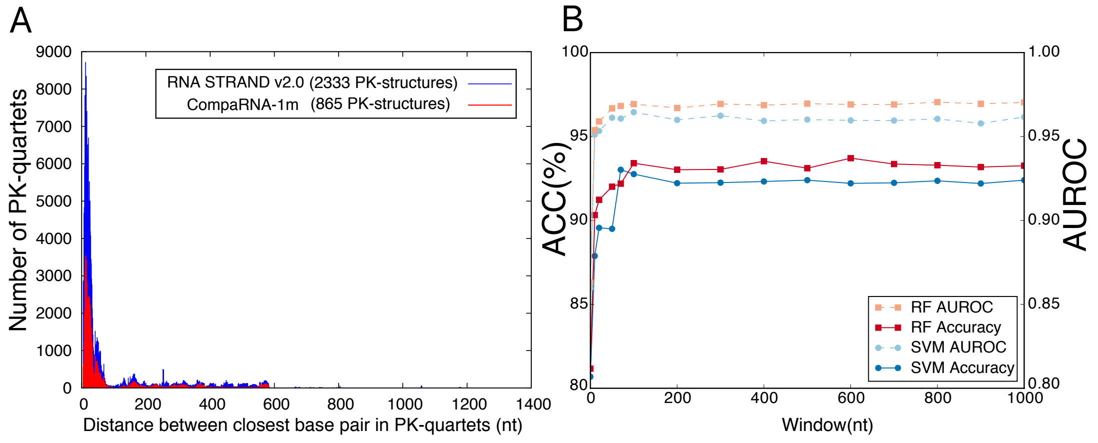


Figure 3.2: Analysis of window size in the feature calculations for the annotated PK-quartets in CompaRNA-1m. **A.** The histograms of the distance between the closest nucleotides participating in base pairs in annotated PK-quartets in RNA STRAND v2.0 and CompaRNA-1m demonstrate that the vast majority have base pairs involving nucleotides within 100nt of each other. **B.** The accuracy (ACC) and AUROC values using our best models, Random Forest (RF) and Support Vector Machine (SVM), exhibit trends with window size w consistent with the distribution of distances observed in CompaRNA-1m. The analysis is done with 100 iterations for each window size using leave-p-out cross validation (p=10%). As is shown, both ACC and AUROC increase as the window size changes from 0,10,20,50,70 up to 100 and plateau after 100 nt window size.

Not content with these basic sequence features, we sought to define features that provide more information about the possible structures that could be formed. We can define a set of unordered pairs $\mathbb{P} = \{\{A, U\}, \{C, G\}, \{G, U\}\}$ as the set of allowable base pairs. An RNA secondary structure S can be described as a set of ordered pairs (i, j) such that $1 \leq i < j \leq n$, corresponding to positions of paired nucleotides with $\{x[i], x[j]\} \in \mathbb{P}$. A structure can be described as a PKF-structure if for all base pairs (i, j) and $(i', j') \in S$ where $i < i'$, it is such that $i < i' < j' < j$ or $i < j < i' < j'$. Alternatively, a structure is described as a PK-structure if there exist two pairs such that $i < i' < j < j'$. Fig 3.1 depicts these two situations. We refer to a quartet of positions i, j, i', j' as a "PK-quartet" if it consists of two base pairs that satisfy the PK ordering, and as a "PKF-quartet" for those that satisfy the PKF ordering. It is useful to define the notation for the pair of nucleotides at positions i and j by $x_{ij} = \{x[i], x[j]\}$. In what follows, we will compute quantities relative to positions i and j when $x_{ij} \in \mathbb{P}$.

We can quantify the propensity for the formation of PK- and PKF-quartets by an aggregate score that counts the number of possible PK and PKF base pairs. The reasoning is that PK-quartets are more likely if there are more ways for PK-quartets to happen. To compute this score, we first define a count matrix $C_{i,b}$ that quantifies the number of occurrences of nucleotide b in $x[1..i]$. After applying the base case $C_{0,b} = 0$ for all b, the terms of this matrix can be computed in linear time by the recurrence relation:

$$C_{i,b} = \begin{cases} C_{i-1,b} + 1, & \text{if } x[i] = b \\ C_{i-1,b}, & \text{otherwise} \end{cases}$$

This matrix can be useful for computing the number of occurrences of a nucleotide b inside and outside the positions i and j. For example, we can define the "inner count" as number of occurrences of b in $x[i+1..j-1]$ by $I_{ijb} = C_{j-1,b} - C_{i,b}$. The "outer count", defined as the number of occurrences of b within $x[1...i-1]$ or $x[j+1..n]$, is computed by $O_{ijb} = C_{nb} - C_{jb} + C_{i-1,b}$.

One can compute the number of possible "nested" base pairs either strictly inside or outside x_{ij}, such that they are consistent with PKF base pairs. We can quantify such a count for a particular base pair $\beta = \{\beta_1, \beta_2\} \in \mathbb{P}$ by

$$N_{ij\beta} = I_{ij\beta_1} \times I_{ij\beta_2} + O_{ij\beta_1} \times O_{ij\beta_2}$$

Similarly, we can define counts for the number of possible "(pseudo)knotted" base pairs overlapping a set of base pairs i and j, consistent with PK base pairs.

$$K_{ij\beta} = I_{ij\beta_1} \times O_{ij\beta_2} + I_{ij\beta_2} \times O_{ij\beta_1}$$

Clearly, both equations are not changed by swapping β_1 and β_2, consistent with the notion that β is an unordered pair. We can then define $2|\mathbb{P}|^2 = 18$ features corresponding to both nested and knotted base pair counts for two base pairs α and β. These scores can be computed for PKF-base pairs as

$$PKF_{\alpha\beta} = \sum_{i=1}^{n} \sum_{j=i+d_{min}}^{i+w} \delta_{\alpha x_{ij}} N_{ij\beta} W_{\alpha} W_{\beta}$$

and for PK-base pairs as

$$PK_{\alpha\beta} = \sum_{i=1}^{n} \sum_{j=i+d_{min}}^{i+w} \delta_{\alpha x_{ij}} K_{ij\beta} W_{\alpha} W_{\beta} \tag{3.1}$$

Where d_{min} provides the minimum distance allowed between base pairs (default = 3), and $\delta_{\alpha x_{ij}}$ is the Kronecker delta that will pick out appropriate base pairs is defined as

$$\delta_{\alpha x_{ij}} = \begin{cases} 1, & \text{if } x_{ij} = \alpha \\ 0, & \text{if } x_{ij} \neq \alpha \end{cases}$$

The terms W_{α} are optional weights that quantify the strength of base pairs for each $\alpha \in \mathbb{P}$. This score reflects the pairing stability, so we chose values as the approximate negative energy, so $W_{\{C,G\}} = 3$, $W_{\{A,U\}} = 2$, and $W_{\{G,U\}} = 1$ [32]. The quantities $PK_{\alpha\beta}$ and $PKF_{\alpha\beta}$ correspond to 18 total features. These features give an estimate of the number of ways that the given RNA can form PK- and PKF-quartets using base pairs α and β. Note the use of the window w. We are able to reduce the complexity for computing $PK_{\alpha\beta}$ and $PKF_{\alpha\beta}$ from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \times w)$. This is under the hypothesis that at least one of the base pairs participating in any PK-quartets would have nucleotides within some distance w along the transcript. To verify this hypothesis, we examined the minimum distance between closest base pairs in PK-quartets using both RNA STRAND v2.0, CompaRNA-1m databases (Fig 3.2A). We found that the closest base pair of 78.3% of PK-quartets consisted of nucleotides less than 100 nt apart, and 99.3% are less than 600 nt apart for CompaRNA-1m. Furthermore, to investigate how the window size affects the performance of our model, we plotted the accuracy values with respect to each window size for two of our best ML models (See "Model Selection") in Fig 3.2B. As is shown in the figure, changing the window size after 100 nt does not significantly affect the accuracy. However, given the distribution presented in Fig 3.2A, we chose the window size of 100 nt to include most of the observed PK-quartets while still being fast. Clearly, sequences less than or equal to w will produce an $\mathcal{O}(n^2)$ calculation, but these sequences will be short enough to make this not formidable. For sequences substantially greater than 100 nt, in our case approximately 70%, this calculation is $\mathcal{O}(n)$.

We recognize that $PK_{\alpha\beta}$ and $PK_{\beta\alpha}$ are both estimating the same quantity, the number of possible PK-quartets consisting of base pairs α and β in the sequence. However, our use of the window leads to an asymmetry, resulting in these quantities being different. Therefore, given the reduced computational complexity afforded by the windowing, we have distinct features for $PK_{\alpha\beta}$ and $PK_{\beta\alpha}$, and similarly for $PKF_{\alpha\beta}$ and $PKF_{\beta\alpha}$.

Taken together, we refer to these features as “PK-features” and “PKF-features”. In addition, we also included a term P_{PK}, the “PK-proportion”, defined as the proportion of counts for PK-feature found within all counts for both PK- and PK-features, and given by the equation

$$P_{PK} = \frac{\sum_{\alpha,\beta} PK_{\alpha\beta}}{\sum_{\alpha,\beta} PK_{\alpha\beta} + PKF_{\alpha\beta}}.$$

Similarly, we define P_{PKF} as the “PKF-proportion”. Collectively, we can refer to all of these proportions and features as “BP-features”, because they consist of potential base pairs that may or may not participate in PK-quartets. Figure 3.2B not only serves to motivate the choice of the window size w, but it also serves to demonstrate the added benefit of the BP-features compared to other features. For example, the accuracy for RF goes from 81.17% for $w = 0$ to 93.41% for $w = 100$ nt.

In addition, we explored considering dinucleotide pairs, such as seen in the nearest neighbor model [49], and features that analogously count the propensity of dimers to pair with complementary dinucleotides, “dimer-PK-features” and “dimer-PKF-features”. For example, for dimer pairs ϕ and ψ, such as “`AC:GU`” and “`GC:GC`”, we can define $dPK_{\phi\psi}$ analogously to Equation 3.1, with weights defined as nearest neighbor energy contributions [106]. These “dimer-features” did not significantly improve the performance, likely due to the large number (968) of features relative to the size of our training set. However, we do include “dimer-PK-proportion”, P_{dPK}, and “dimer-PKF-proportion”, P_{dPKF}, defined analogously as P_{PK} and P_{PKF} as features in the output. These quantities can be computed in $\mathcal{O}(nw)$, similarly to the BP-features.

Overall, we extracted 103 features from each RNA sequence for use in PK-detection.

3.2.3 Model Selection

To train and validate the extracted features, four supervised learning methods were selected for classification: Random Forest (RF), Support Vector Machine (SVM), Adaptive Boosting (AB), and Logistic Regression (LR). We evaluated each of the models using 100 iterations, with PK defined as the positive class, and collected accuracy and generalization error for each iteration. To avoid overfitting, leave-p-out (LPO) and 10-fold (10x) [135] cross validation strategies were used for splitting the data into training and testing sets for each iteration. In our LPO cross validation, we used $p = 10\%$ so that during each iteration 10% of the data are randomly selected for testing and the rest is considered as a training set. Whereas, in 10x cross validation, the data are randomly partitioned into 10 equal sizes (9 sections for training, 1 for testing), and likewise for each combination of 9 and 1. In these cross validation strategies each subset is selected to be either for training or testing for a given iteration, ensuring that we never test on the same data used to train the model. For our machine learning models, we used implementations from the `scikit-learn` package.

The first model considered was an ensemble learning method called Random Forest (RF) [22]. Through training time, RF generates n_{tree} decision trees based on a randomly selected subset of m_{try} features. After training, the output prediction is the average prediction over the n_{tree} decision trees. The RF algorithm overcomes the overfitting problem, since it averages over multiple decision trees and reduces the variance of the classifier [19]. We used the default value of $n_{tree} = 300$ and heuristically chose the best value for m_{try} by minimizing the training error, giving the optimal value of 32.

The second model considered is Support Vector Machine (SVM). SVM is a supervised learning algorithm that divides the training set into separate classes (in our problem two classes) and later assigns the testing samples to a relative class [38]. SVM with a kernel can map the data to a space where the data are linearly separable with a non-probabilistic binary classifier. We found that the kernel with the best performance was with the Gaussian or radial basis function (RBF) [28]. To find the best parameters for SVM, we used `GridSearchCV` from `scikit-learn` and determined the best parameters were gamma=0.01 and c=10 using the RFB kernel.

The third model, Adaptive Boosting (AB), is a practical and efficient boosting algorithm that invokes weak learners many times on different distributions over the training data [50]. Simply, in machine learning the algorithms that perform slightly better than random guessing are called weak learners, and AB combines weak learners to construct a strong learner by adding a new weak learner in every iteration during the training. The result is a strong classifier with higher accuracy since in every iteration the focus is on the mis-classified examples and hence the error in each step will be used as a focus for the next iteration over the new weighted training data. AB has the ability to overcome overfitting most of the time by the use of early stopping, but is sensitive to noise and outliers in the data. However, AB is still a good candidate for our work since it is robust and efficient. As far as the weak learners are concerned, we used the common case of decision tree learners.

Finally, we considered logistic regression (LR) model [20]. LR has the advantage of being a deterministic model with weight parameters that can be easily interpreted. The LR model computes weights by maximizing the likelihood estimation over the training set using the extracted features.

Table 3.1: Average performance of different machine learning algorithms on PK-detection.

Model	**Mean ACC**	**Median ACC**	**Median AUROC**	**Mean Sens**	**Mean Spec**	**Mean PPV**
$RF - LPO$	93.09	94.05	97.67	91.07	95.34	95.24
$RF - 10x$	92.06	93.18	97.84	91.66	94.74	94.84
$SVM - LPO$	91.96	93.06	96.94	93.10	92.59	92.72
$SVM - 10x$	91.90	92.35	97.21	93.21	92.77	93.10
$AB - LPO$	90.41	92.07	95.89	89.80	93.18	93.75
$AB - 10x$	91.47	90.34	95.95	90.24	92.63	92.68
$LR - LPO$	86.60	87.13	93.24	86.53	88.23	88.46
$LR - 10x$	86.13	86.93	93.75	86.59	87.91	87.78

The average classification accuracy (ACC), AUROC, sensitivity (Sens), specificity (Spec), and positive predicted value (PPV) for four different machine learning algorithms on 100 iterations using leave-p-out (LPO) and 10-Fold (10x) cross validation on CompaRNA-1m demonstrate that RF and SVM perform best compared to other models.

3.2.4 Pseudoknow-Assisted Folding

We next determined if `Pseudoknow` could help improve secondary structure prediction by anticipating whether the input RNA has a PK or not, and informing the choice of the appropriate program accordingly. This method, called Pseudoknow-assisted folding, is depicted in Fig 3.5A. The input RNA sequence is evaluated using `Pseudoknow`, with Leave-One-Out (LOO) cross validation [135], and is triaged according whether it has a PK or not. If the probability of a PK is greater than 0.5, we compute the structure using `IPknot` [140], otherwise, we use `CONTRAfold` [44]. These two particular programs were selected because they were previously determined to have the highest accuracy in non-comparative secondary structure prediction [127]. The program `ContextFold` [186] had higher accuracy compared to `CONTRAfold`, but it is a machine learning program that was trained on RNA STRAND v2.0; hence, we did not use this program because we are testing on structures derived from this data set. For this analysis, we used `IPknot` version 0.0.4 and `ContraFold` version 2.02, both of which are currently the latest versions.

3.2.5 Evaluation Metrics

We consider the following metrics for the purpose of comparison of our results with other existing methods: accuracy (ACC), sensitivity (Sens), specificity (Spec), and positive predicted value (PPV). These equations are all based on the definitions of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). TP is the number of truly classified PKs and FP is the number of incorrectly classified PK-structures in our analysis. Likewise, TN is the number of correctly classified PKF-structures, and FN is the number of PK-structures that are incorrectly identified as PKF-structures. The ACC formulation is the total valid predictions (sum of TP and TN) over the total number of samples.

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

Sensitivity or true positive rate (TPR) measures the proportion of the correctly classified PK-structures:

$$Sens = \frac{TP}{TP + FN}$$

The specificity, or true negative rate (TNR), evaluates the proportion of PKF structures that are correctly identified in our model.

$$Spec = \frac{TN}{TN + FP}$$

The PPV value explains the positive proportions of results over the sum of both true positive (TP) and false positives (FP).

$$PPV = \frac{TP}{TP + FP}$$

For structure prediction, the accuracy, specificity, etc calculations correspond to the prediction of individual base pairs in the annotated structures. The difference is, this calculation involves a much larger TN term, corresponding to the total number of unpaired nucleotides, which tends to dominate many calculations such as accuracy, resulting in very small differences between programs for different software. Quantities such as sensitivity and PPV, which do not include TN, do not suffer from this effect. We estimated the total number of possible base pairs as $N_A \times N_U + N_G \times N_C + N_G \times N_U$. Therefore, we compute TN by

$$TN = N_A \times N_U + N_G \times N_C + N_G \times N_U - TP - FN - FP$$

For secondary structure prediction evaluation, we next introduce the quantity ϵ, the number of compatible false-positive base pairs. As in [127] and [55], we only penalize base pairs as false positives if they are incompatible with the annotated structure. This results in a modified calculation for PPV, given by

$$PPV = \frac{TP}{TP + FP - \epsilon}$$

We also used the Matthews correlation coefficient (MCC) to measure the performance of different methods in the structure prediction. The formulation of MCC is as follows:

$$MCC = \frac{TP \times TN - (FP - \epsilon) \times FN}{\sqrt{(TP + FP - \epsilon) \times (TP + FN) \times (TN + FP - \epsilon) \times (TN + FN)}}$$

3.3 Results and Discussion

3.3.1 PK-detection

Overall, we trained and tested the four models—RF, SVM, AB, and LR—for 100 iterations and the error of each model was captured in every iteration. Fig 3.3 represents the ACC, Sens, Spec, and PPV box plots of the four models in the 100 iterations. In addition, Table

3.1 demonstrates the mean and median values for ACC, as well as for the AUROC, Sens, Spec, and PPV for all ML models, and for both LPO, and 10x cross-validation strategies. Among all these four models, RF has the best performance with 94.05% accuracy on average. SVM is next on the list with 93.06% average accuracy within the 100 iterations.

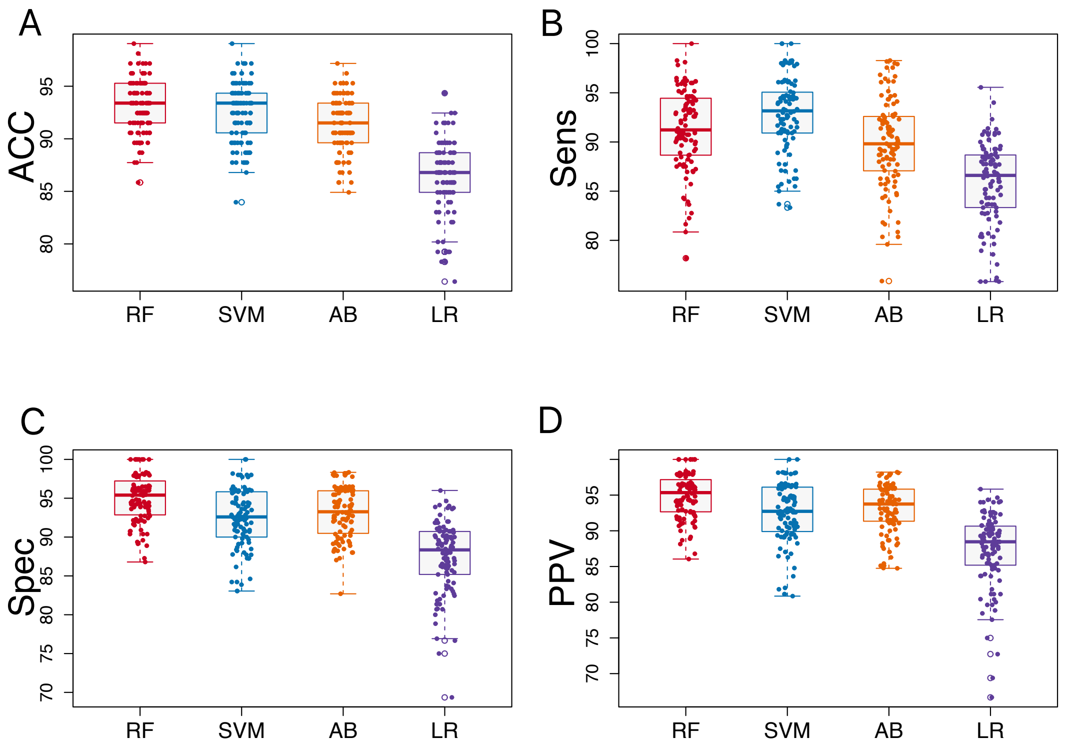


Figure 3.3: Performance results of four different ML models using 100 iterations and LPO cross-validation. Box plots of accuracy (ACC), sensitivity (Sens), specificity (Spec), and positive predicted value (PPV) of LPO cross-validation on 100-iterations. When comparing Random Forest (RF), Support Vector Machine (SVM), Adaptive Boosting (AB), and Logistic Regression (LR), RF and SVM have the highest values on average.

The area under ROC curve (AUROC) value of each model also was calculated through the runs. ROC curve diagram in Fig 3.4A shows the overall performance over all models and demonstrates that RF and SVM have more accurate results compared to the other two methods, while Fig 3.4B demonstrates the AUROC values trends of all models for 100 iterations on average. As is shown, RF has the best AUROC compared to other models, and again SVM is the next best on the list.

To better assess `Pseudoknow`'s performance in PK-detection, we compared it to pseu-

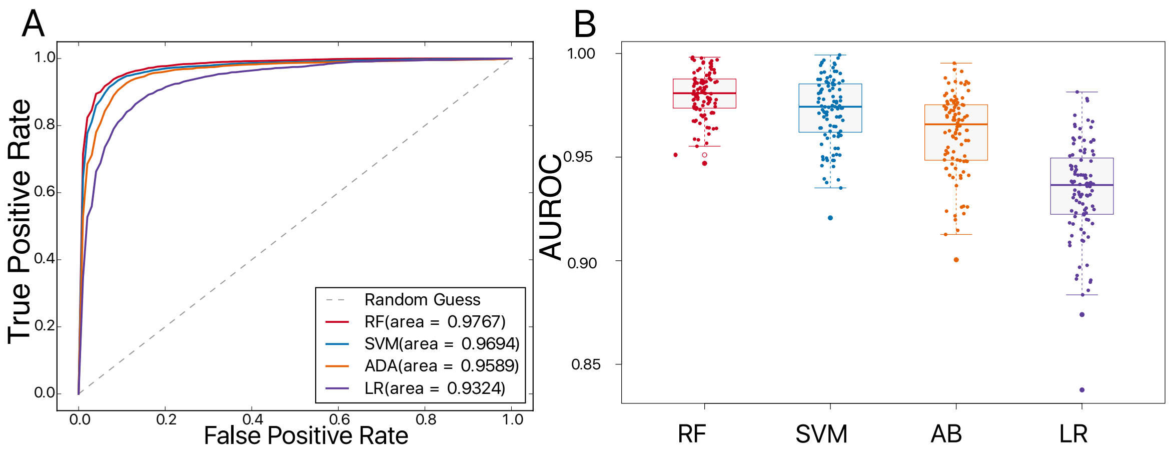


Figure 3.4: ROC curve trend and AUROC box plot of RF, SVM, AB, and LR models. **A.** ROC curve comparison for four different machine learning techniques shows that RF and SVM perform best on average with our feature sets and data. **B.** AUROC boxplots for 100-iterations applied to the four models examined. Among all four, RF and SVM perform best with an average AUROC of 97.67% and 96.94%.

doknot presence or absence in pre-computed secondary structure predictions for `IPknot`, `ProbKnot`, `DotKnot`, `McQFold`, and `pknotsRG` [140, 14, 153, 109, 134], downloaded as part of the CompaRNA data set. Table 3.2 shows a comparison of `Pseudoknow`'s performance in PK-detection compared to programs designed for pseudoknot secondary structure prediction. For this table, we only considered programs that predict non-comparative secondary structure and have no limitations on the input size. `Pseudoknow` performs better in every category of PK-detection performance. For example, the program `IPknot` gives an accuracy of 74.64% when used to detect the presence or absence of PK-structures in CompaRNA-1m using the McCaskill model and default weighting compared to 93.86% for `Pseudoknow`.

Table B.3 shows a comparison of `Pseudoknow`'s performance in PK-detection compared to other software when using different disjoint subsets of CompaRNA-1m for training and testing. While the LPO and 10x cross-validation strategies train and validate on complementary subsets, this result shows how `Pseudoknow` performs when we consider training and validation sets that originate from different sources. Overall, `Pseudoknow` has an average accuracy of 91.71%, whereas the second best method, `IPknot`, has an average accuracy of 76.70%.

Table 3.2: Comparison of the mean performance in PK-detection between `Pseudoknow` and other software.

Software	ACC(%)	Sens(%)	Spec(%)	PPV(%)	MCC
`Pseudoknow`	93.86	92.39	92.40	95.38	0.87
`IPknot`	74.64	77.69	72.00	70.48	0.50
`ProbKnot`	71.72	59.63	82.12	74.15	0.43
`DotKnot`	70.81	84.66	58.90	63.93	0.45
`McQFold`	66.78	49.08	82.02	70.14	0.33
`pknotsRG`	60.64	28.62	88.20	67.61	0.21

`Pseudoknow` performs better in classifying PK-structures compared with existing RNA folding algorithms. The results show that even though these existing programs have high accuracy in structure prediction, they often fail at PK-detection. These values were computed using Leave-One-Out (LOO) cross-validation for `Pseudoknow` in order to compare one sequence at a time.

3.3.2 Improved Secondary Structure Prediction

When predicting secondary structure, researchers often have to choose programs for PK- or PKF-structure prediction. These programs can vary in their sensitivity and specificity in structure prediction. For example, `IPknot` version 0.0.4 predicts structures with pseudoknots, but with lower sensitivity and lower specificity than `CONTRAfold` [44], which does not predict pseudoknots. We found that using `Pseudoknow` to assist in the selection of which software to use based on PK-detection probability, led to improved mean accuracy, specificity, and PPV compared to these two methods. In addition, we found that this approach led to the improvement in sensitivity and MCC compared to the average of these two approaches on CompaRNA-1m. These results are summarized in Table 3.3, and in addition, the median accuracy, specificity etc. is presented in Table B.1

We further evaluated our model using an additional validation set that was completely different from our training set. We downloaded the PDB data set from the CompaRNA server (`http://genesilico.pl/comparna/`), hereafter “PDB-validation”, consisting of the 342 RNA structures deposited to PDB between February 2009 and February 2016. Note that our validation set “PDB-validation” is different from the data set “PDB”, which is part of CompaRNA-1m. We applied a similar approach of filtering to this validation set by removing redundant structures using `CD-HIT-EST`. Many of the structures in this data set

Table 3.3: Secondary structure prediction results on CompaRNA-1m.

Software	ACC(%)	Sens(%)	Spec(%)	PPV(%)	MCC
`Pseudoknow`	98.34	59.69	98.75	74.65	0.658
`IPknot`	93.02	49.13	93.54	69.50	0.575
`Contrafold`	98.20	64.93	98.59	71.52	0.676

`Pseudoknow` as a preliminary step for structure prediction improves the RNA folding of two well-known programs, `IPknot` and `CONTRAfold`, on CompaRNA-1m. Pseudoknow-assisted folding has higher mean accuracy, specificity, and PPV compared to these programs on their own. While `ContraFold` has higher mean sensitivity and MCC, it can not predict pseudoknots. `Pseudoknow` predicts pseudoknots and also has higher mean sensitivity and MCC than the average of the other methods.

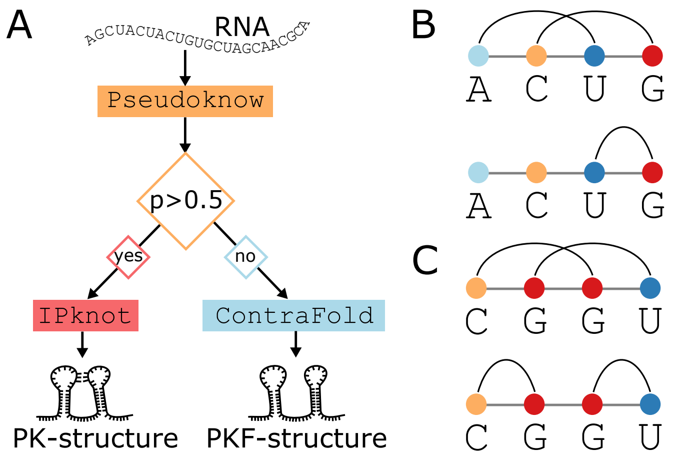


Figure 3.5: Pseudoknow-assisted folding representation chart and examples of base pairing possibilities in two cases of definite and indefinite**A.** A simple flow diagram of RNA secondary structure prediction with the help of `Pseudoknow`. From an input sequence of an RNA, the features feed into the ML model and, based on the PK-probability, the appropriate software will predict the structures.**B.** Definite quartets: When all nucleotides are paired, the pairs AU and CG can only form PK-quartets. In this example, AU-CG is an example of definite pairings. **C.** Indefinite quartets: When all nucleotides are paired, the pairs CG and GU can form both PK-quartets and PKF-quartets.

consist of RNAs in complex with proteins and other RNAs that could affect the structure. Since none of the software we are evaluating can take this into consideration, we removed all structures for RNAs in molecular complexes. Furthermore, because many structures

Table 3.4: Secondary structure prediction results on an independent validation set, PDB-validation.

Software	**ACC(%)**	**Sens(%)**	**Spec(%)**	**PPV(%)**	**MCC**
`Pseudoknow`	99.60	88.81	99.90	95.35	0.916
`IPknot`	99.44	85.63	99.92	95.22	0.898
`Contrafold`	99.55	88.74	99.88	93.74	0.907

Pseudoknow-assisted folding improves RNA structure prediction for the 63 RNA structures in the PDB-validation set. We improved mean accuracy, sensitivity, PPV, and MCC compared to `IPknot` and `CONTRAfold`. Also, Pseudoknow-assisted folding produces better mean specificity than the average of these two algorithms.

change conformation when bound to a ligand, including pseudoknots that are stabilized by the ligand binding [58], we also removed structures of RNAs bound to ligands. The final set in our validation consists of 63 RNA structures, of which 16 are PKs.

Table 3.4 and Table B.2 show the mean/median performance evaluation of this triage method for secondary structure prediction using PDB-validation. For this data set, `ContraFold` has higher sensitivity than `IPknot`, whereas `IPknot` has higher specificity. In comparison, Pseudoknow-assisted folding has higher mean accuracy, sensitivity, PPV, and MCC than either method on its own. The specificity of Pseudoknow-assisted folding is also higher than the average of `IPknot` and `CONTRAfold`. These results are certainly in part due to the fact that `Pseudoknow` does a better job in PK-detection for the PDB-validation set than `IPknot`. Taken together, these data suggest that, by anticipating the presence of pseudoknots, `Pseudoknow` can lead to a greater balance of sensitivity and specificity, resulting in improved performance.

Besides having a powerful predictive model for classifying PK-structures, identifying which features are most powerful could potentially shed light on pseudoknot formation. Accordingly, we used the Boruta algorithm [86] to identify the importance of each feature, which is a wrapper algorithm around random forest classification that systematically removes features and quantifies the corresponding reduction in classification performance. In the end, this procedure can identify the most relevant features for classification. The feature selection analysis shows the most important and relevant features in green, less relevant in yellow and red (Figure B.3). Most of our features are found to be relevant (85 out of 103).

The trimer frequency for "UGG" is the most important feature produced from the Boruta analysis. Analyzing both structures and sequences in CompaRNA-1m, we found that UGG is significantly enriched in the sequences that form PKF-structure. In addition, we found that among trimers that are completely base-paired, UGG is the most enriched in PKF-structures (see Figure B.4). The dimer frequency for "UG" is the fourth most important feature, and also was found to be enriched in sequences that form PKF-structures compared to sequences that form PK-structures (see Figure B.5).

The second and third most important features in our list are the BP-features "$PK_{AU,CG}$" and "$PKF_{AU,CG}$". To interpret why these two features are most important among other BP-features, we introduced two categories of nucleotide quartets, "definite" and "indefinite". In the definite case, the situations where all four nucleotides are base-paired always produce either nested pairings or pseudoknotted pairings, while in an indefinite case, nucleotides have the ability to participate in both PK- and PKF-quartets (Fig 3.5B-C). After this analysis, we concluded that the importance of BP-features with AU-CG quartets in PK-detection could be due to the fact that it is the only definite nucleotide quartet.

3.4 Summary and Conclusions

Our machine learning approach quickly classifies a given RNA as being likely to have pseudoknot structures with 94% accuracy. We have demonstrated the potential utility of this method for genome-wide structure analyses, in that it could enable an efficient grouping of RNAs into PK and PKF categories, thus informing which type of structure prediction to subsequently perform. Moreover, it can group RNAs into these categories more accurately than the best non-comparative RNA folding methods. Because of this, `Pseudoknow` is a valuable utility as a first step in secondary structure prediction analysis.

We have defined 103 features, including our novel BP-features, for input to our ML model. In general, the more data available for training ML models, the better they perform in classification. We expect the performance of the `Pseudoknow` to improve as more experimentally determined secondary structures become available for training. With the growing interest in RNA structure and function, we expect these data to become available in the near future, leading to further improvements to PK-detection and structure prediction with `Pseudoknow`.

Chapter 4: An Interpretable Deep Learning Approach for Cancer Detection and Relevant Gene Identification [1]

4.1 Introduction

The analysis of gene expression data has the potential to lead to significant biological discoveries. Much of the work on the identification of differentially expressed genes has focused on the most significant changes, and may not allow recognition of more subtle patterns in the data [77, 115, 181, 92, 187, 113]. Tremendous potential exists for computational methods to analyze these data for the discovery of gene regulatory targets, disease diagnosis and drug development [103, 141, 146]. However, the high dimension and noise associated with these data presents a challenge for these tasks. Moreover, the mismatch between the large number of genes and typically small number of samples presents the challenge of a "dimensionality curse". Multiple algorithms have been used to distinguish normal cells from abnormal cells using gene expression [132, 51, 108, 161]. Although there has been a lot of research into cancer detection from gene expression data, there remains a critical need to improve accuracy, and to identify genes that play important roles in cancer.

Machine learning methods for dimensionality reduction and classification of gene expression data have achieved some success, but there are limitations in the interpretation of the most significant signals for classification purposes [40, 83]. Recently, there have been efforts to use single-layer, nonlinear dimensionality reduction techniques to classify samples based on gene expression data [162]. In similar studies of computer vision, unsupervised deep learning methods have been successfully applied to extract information from high dimensional image data [89]. Similarly, one can extract the meaningful part of the expression data by applying such techniques, thereby enabling identification of specific subsets of genes that are useful for biologists and physicians, with the potential to inform therapeutic strategies.

In this work, we used stacked denoising autoencoders (SDAE) to transform high-dimensional, noisy gene expression data to a lower dimensional, meaningful representation

[1] A version of this chapter has been published [Danaee, Padideh, Reza Ghaeini, and David A. Hendrix. PACIFIC SYMPOSIUM ON BIOCOMPUTING, 2017]

[173]. We then used the new representations to classify breast cancer samples from the healthy control samples. We used different machine learning (ML) architectures to observe how the new compact features can be effective for a classification task and allow the evaluation of the performance of different models. Finally, we analyzed the lower-dimensional representations by mapping back to the original data to discover highly relevant genes that could play critical roles and serve as clinical biomarkers for cancer diagnosis. The performance of these methods affirm that SDAEs could be applied to cancer detection in order to improve the classification performance, extract both linear and nonlinear relationships in the data, and perhaps more important, to extract a subset of relevant genes from deep models as a set of potential cancer biomarkers. The identification of these relevant genes deserves further analysis as it potentially can improve methods for cancer diagnosis and treatment.

Classification and clustering of gene expression in the form of microarray or RNA-seq data are well studied. There are various approaches for the classification of cancer cells and healthy cells using gene expression profiles and supervised learning models. The self-organizing map (SOM) was used to analyze leukemia cancer cells [59]. A support vector machine (SVM) with a dot product kernel has been applied to the diagnosis of ovarian, leukemia, and colon cancers [51]. SVMs with nonlinear kernels (polynomial and Gaussian) were also used for classification of breast cancer tissues from microarray data [132].

Unsupervised learning techniques are capable of finding global patterns in gene expression data. Gene clustering represents various groups of similar genes based on similar expression patterns. Hierarchical clustering and maximal margin linear programming are examples of this learning and they have been used to classify colon cancer cells [4, 93]. K-nearest neighbors (KNN) unsupervised learning also has been applied to breast cancer data [108].

Due to the large number of genes, high amount of noise in the gene expression data, and also the complexity of biological networks, there is a need to deeply analyze the raw data and exploit the important subsets of genes. Regarding this matter, other techniques such as principal component analysis (PCA) have been proposed for dimensionality reduction of expression profiles to aid clustering of the relevant genes in a context of expression profiles [185]. PCA uses an orthogonal transformation to map high dimensional data to linearly uncorrelated components [177]. However, PCA reduces the dimensionality of the data linearly and it may not extract some nonlinear relationships of the data [61]. In contrast, other approaches such as kernel PCA (KPCA) may be capable of uncovering these nonlinear relationships [142].

Similarly, researchers have applied PCA to a set of combined genes of 13 data sets to obtain the linear representation of the gene expression and then apply a autoencoder to capture nonlinear relationships [47]. Recently, a denoising autoencoder has been applied to extract a feature set from breast cancer data [162]. Using a single autoencoder may not extract all the useful representations from the noisy, complex, and high-dimensional expression data. However, by reducing the dimensionality incrementally, the multi-layered architecture of an SDAE may extract meaningful patterns in these data with reduced loss of information [15].

4.2 Materials and Methods

We have applied a deep learning approach that extracts the important gene expression relationships using SDAE. After training the SDAE, we selected a layer that has both low-dimension and low validation error compared to other encoder stacks using a validation data set independent of both our training and test set [64]. As a result, we selected an SDAE with four layers of dimensions of 15,000, 10,000, 2,000, and 500. Consequently we used the selected layer as input features to the classification algorithms. The goal of our model is extracting a mapping that possibly decodes the original data as closely as possible without losing significant gene patterns.

We evaluated our approach for feature selection by feeding the SDAE-encoded features to a shallow artificial neural network (ANN) [174] and an SVM model [37]. Furthermore, we applied a similar approach with PCA and KPCA as a comparison.

Lastly, we used the SDAE weights from each layer to extract genes with strongly propagated influence on the reduced-dimension SDAE-encoding. These selected "deeply connected genes" (DCGs) are further tested and analyzed for pathway and Gene Ontology (GO) enrichment. The results from our analysis showed that in fact our approach can reveal a set of biomarkers for the purpose of cancer diagnosis. The details of our method are discussed in the following subsections, and the work-flow of our approach is shown in Figure C.1.

4.2.1 Gene Expression Data

For our analysis, we analyzed RNA-seq expression data from The Cancer Genome Atlas (TCGA) database for both tumor and healthy breast samples [176]. These data consist of 1097 breast cancer samples, and 113 healthy samples. To overcome the class imbalance of the data, we used synthetic minority over-sampling technique (SMOTE) to transform data into a more balanced representation for pre-training [30]. We used the `imbalanced-learn` package for this transformation of the training data [90]. Furthermore, we removed all genes that had zero expression across all samples.

4.2.2 Dimensionality Reduction Using Stacked Denoising Autoencoder

An autoencoder (AE) is a feedforward neural network that produces the output layer as close as possible to its input layer using a lower dimensional representation (hidden layer). The autoencoder consists of an encoder and a decoder. The encoder is a nonlinear function, like a sigmoid, applied to an affine mapping of the input layer, which can be expressed as $f_\theta(X) = \sigma(Wx + b)$ with parameters $\theta = \{W, b\}$. The matrix W is of dimensions $d' \times d$ to go from a larger dimension of gene expression data d to a lower dimensional encoding corresponding to d'. The bias vector b is of dimension d'. This input layer encodes the data to generate a hidden or latent layer. The decoder takes the hidden representations from the previous layer and decodes the data as closely as possible to the original inputs, and can be expressed as $z = g_{\theta'}(y) = \sigma(W'y + b')$. In our implementation, we imposed tied weights, with $W' = W^T$. We can refer to the weight matrix W and bias b as $\theta = \{W, b\}$ and similarly $\theta' = \{W', b'\}$.

A SDAE can be constructed as a series of AE mappings with parameters $\theta_1, \theta_2, ..., \theta_n$ and the addition of noise to prevent overfitting [173]. In order to get a good representation for each layer, we maximize the information gain between the input layer (modeled as a random variable X from an unknown distribution $q(X)$) and its higher level stochastic representation (random variable Y from a known distribution $p(X|Y;\theta')$). For layer i, we then learned a set of parameters θ_i and θ_i' from a known distribution p where $q(Y|X) = p(Y|X;\theta_i)$ and also $q(X|Y) = p(X|Y;\theta_i')$ that maximize the mutual information [173].

This maximization problem corresponds to minimizing the reconstruction error of the input layer using hidden representation. In this construction, the hidden layer contains the compressed information of the data by ignoring useless and noisy features. In fact, the autoencoder extracts a set of new representations that encompass the complex relationships between input variables. The reconstruction error of the input layer using this new representation is non-zero, but can be minimized. In practice, the weights of the model are learned through the stochastic gradient descent (SGD) algorithm [138, 21].

Autoencoders extract both linear and nonlinear relationships inherent in the input data, making them powerful and versatile. The encoder of the SDAE decreases the dimensionality of the gene expression data stack-by-stack, which leads to reduced loss of information compared to reducing the dimension in one step [15]. In contrast, the decoder increases the dimensionality to eventually achieve the full reconstruction of the original input as close as possible. In this procedure, the output of one layer is the input to the next layer. For this implementation, we used the `Keras` library with `Theano` backend running on an Nvidia Tesla K80 GPU [33]. Although it is difficult to estimate the time complexity of the deep architecture of the SDAE, with batch training and highly parallelizable implementation on GPUs, training takes a few minutes and testing of a sample is performed in a few seconds.

It is proven in practice that pre-training the parameters in a deep architecture leads to a better generalization on a specific task of interest [173]. Greedy layer-wise pre-training is an unsupervised approach that helps the model initialize the parameters near a good local minimum and convert the problem to a better form of optimization [15]. Therefore, we considered the pre-training approach as supposed to achieve smoother convergence and higher overall performance in cancer classification. After starting with the initial parameters resulting from the pre-training phase, we used supervised fine-tuning on the full training set to update the parameters.

To avoid overfitting in the learning phase (both pre-training and fine-tuning) of the SDAE, we utilized a dropout regularization factor, which is a method of randomly excluding fractions of hidden units in the training procedure by setting them to zero. This method prevents nodes from co-adapting too much and consequently avoids overfitting [155]. For the same purpose, we provided partially corrupted input values to the SDAE (denoising). The SDAE is robust, and its accuracy does not change upon introducing noise at a low rate. In fact, SDAE with denoising and dropout can find a better representation from the noisy data. Figure 4.1 shows the SDAE encoded, decoded, and denoised representations on the subset of genes.

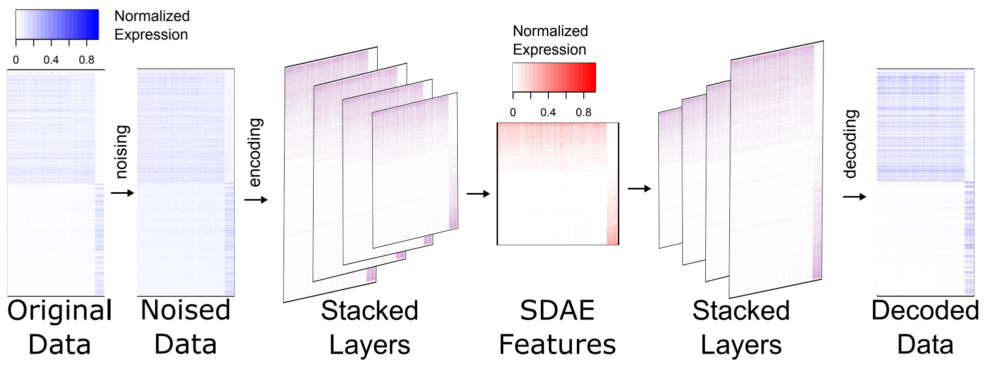


Figure 4.1: SDAE representation using the enriched genes in the TCGA breast cancer. In this depiction for illustrative purposes, the top 500 genes with median expression across cancer samples enriched above health samples, and the top 500 genes with reduced median expression across cancer samples is shown.

4.2.3 Differentially Expressed Genes

We used significantly differentially expressed genes as a comparison to our SDAE features for cancer classification. First, we computed the log fold change comparing the median expression in cancer tissue samples to that of healthy tissue samples. We then computed a two-tailed p-value using a Gaussian fit, followed by a Benjamini-Hochberg (BH) correction [16]. We identified two sets of differentially expressed genes. The first, DIFFEXP0.05 was the 206 genes, 98 up-regulated and 118 down-regulated, that were significant at an FDR of 0.05. The second set, DIFFEXP500, contains the top 500 most significant differentially expressed genes (the same dimension as the SDAE features) using the same 2-tailed p-values, containing 244 up-regulated and 256 down-regulated genes.

4.2.4 Dimensionality Reduction Using Principal Component Analysis

As a second level of comparison, we extracted features using linear PCA to provide a baseline for the performance of linear dimensionality reduction algorithms for our ML models. The same reduced dimensionality of 500 was used. In addition, we used KPCA with an RBF kernel to extract features that by default are of the same dimension as the number of training input samples. For both PCA and KPCA we used an implementation in the `scikit-learn` package [121].

4.3 Results and Discussion

4.3.1 Classification Learning

In order to evaluate the effectiveness of our autoencoder-extracted features, we used two different supervised learning models to classify cancer samples from healthy control samples. First, we considered a single-layer ANN with input nodes directly connected to output layers without any hidden units. If we consider the input units as $X = (x_1, x_2, ..., x_n)$, the output values are calculated as $y = \sigma(\sum_i w_i x_i + b)$. Second, we considered both an SVM with a linear kernel and with a radial basis function kernel (SVM-RBF). We applied 5-fold cross-validation for to exhaustively split the data into train and test sets to estimate the accuracy of each model without overfitting. In each split, the model was trained on 4 partitions and tested on the 5th, ensuring that training and testing are performed on non-overlapping subsets.

4.3.2 Comparison of Different Models

To assess the effectiveness of the SDAE features, we compared their performance in classification to differentially expressed genes and to principal components for different machine learning models. The performance of the SDAE features for classification is summarized in Table 4.1. The best method varies depending on the performance metric, but on these data the SDAE features performed best on three of the five metrics we considered. The highest accuracy was attained using SDAE features applied to SVM-RBF classification. This method also had the highest F-measure. The highest sensitivity was found for SDAE features as well, but using the ANN classification model. KPCA features applied to an SVM-RBF had higher specificity and precision.

4.3.3 Deep Feature Extraction and Deeply Connected Genes

Going beyond classification, there is potential biological significance in understanding what subsets of genes are involved in the new feature space that makes it an effective set for the cancer detection. Previous work on cancer detection using a single-layer autoencoder has evaluated the importance of each hidden node [162]. Here, we analyzed the importance of genes by considering combined effect of each stack of the deep architecture. To extract these genes, we utilized a strategy of computing the product of the weight matrices for

Features	Model	ACC	SENS	SPEC	PPV	F-measure
SDAE	ANN	96.95	**98.73**	95.29	95.42	0.970
	SVM	98.04	97.21	99.11	99.17	0.981
	SVM-RBF	**98.26**	97.61	99.11	99.17	**0.983**
DIFFEXP (500)	ANN	63.04	60.56	70.76	84.58	0.704
	SVM	57.83	64.06	46.43	70.42	0.618
	SVM-RBF	77.391	86.69	71.29	67.08	0.755
DIFFEXP (0.05)	ANN	59.93	59.93	69.95	84.58	0.701
	SVM	68.70	82.73	57.5	65.04	0.637
	SVM-RBF	76.96	87.56	70.48	65.42	0.747
PCA	ANN	96.52	98.38	95.10	95.00	0.965
	SVM	96.30	94.58	98.61	98.75	0.965
	SVM-RBF	89.13	83.31	99.47	99.58	0.906
KPCA	ANN	97.39	96.02	99.10	99.17	0.975
	SVM	97.17	96.38	98.20	98.33	0.973
	SVM-RBF	97.32	89.92	**99.52**	**99.58**	0.943

Table 4.1: Comparison of different feature sets using three classification learning models.

each layer of our SDAE. The result is a $500 \times G$ dimensional matrix W, where G is the number of genes in the expression data, computed for an n-layer SDAE by

$$W = \prod_{i=1}^{n} W_i.$$

Although the weights of each layer of the SDAE are computed with a nonlinear model, the matrix W is a linearization of the compounded effect of each gene on the SDAE features. Genes with the largest weights in W are the most strongly connected to the extracted and highly predictive features, so we called these genes DCGs. We found that the terms of matrix W were strongly normally distributed (Figure 4.2). We identified the subset of genes with the most statistically significant impact on the encoding by fitting the distribution of these values in W to a normal distribution, computing a p-value using this fit,and applying a BH correction with an FDR of 0.05.

4.3.4 Gene Ontology

We examined the functional enrichment of the DCGs through a GO term and Panther pathway analysis. Table C.1 presents the statistically-enriched GO terms under "biological

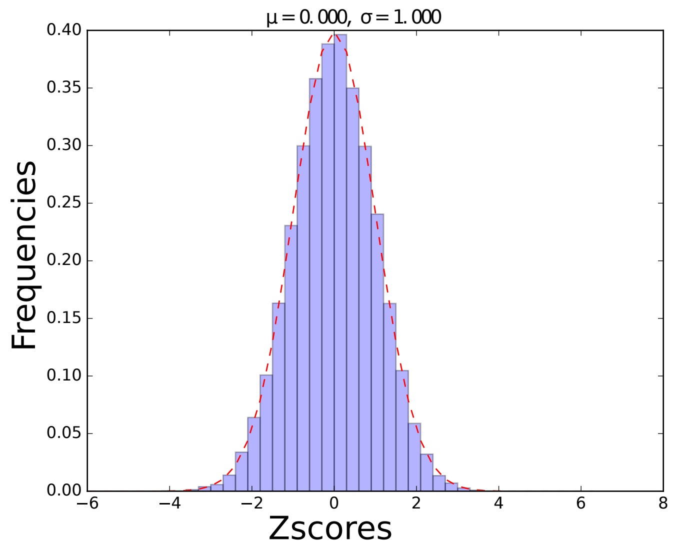


Figure 4.2: Histogram of z-Scores from the dot product matrix of the weights connectivity of the SDAE.

process", and having a Bonferroni-corrected p-value of less than 1e-10. Many of the most significant terms are related to mitosis, suggesting a large number of genes with core functionality that is relevant to cell proliferation. In addition, an analysis of the enrichment of Panther pathways led to a single enriched term, p53 pathway, where we observe 10 genes when 1.34 are expected, giving a p-value of 2.21E-04. P53 is known to be an important tumor-suppressor gene [107, 68, 76] , and this finding suggests a role of tumor suppressor function in many of the DCGs.

4.3.5 Classification Learning Using Deeply Connected Genes

Finally, we used the expression of the DCGs as features for the ML models previously mentioned. These genes served as useful features for cancer classification, achieving 94.78% accuracy (Table 4.2). Although these features performed a few percentage points below that of the SDAE features, they still have advantage of being more readily interpreted. Future work is needed to improve the extraction of DCGs to enhance their utility as features for classification.

Features	Model	ACC	SENS	SPEC	PPV	F-measure
DCGs	ANN	91.74	98.13	87.15	85.83	0.913
	SVM	91.74	88.80	97.50	97.25	0.927
	SVM-RBF	94.78	93.04	97.5	97.20	0.951

Table 4.2: Cancer classification results using deeply connected genes (DCGs).

4.4 Summary and Conclusion

In conclusion, we have used a deep architecture, SDAE, for the extraction of meaningful features from gene expression data that enable the classification of cancer cells. We were able to use the weights of this model to extract genes that were also useful for cancer prediction, and have potential as biomarkers or therapeutic targets.

One limitation of deep learning approaches is the requirement for large data sets, which may not be available for cancer tissues. We expect that as more gene expression data becomes available, this model will improve in performance and reveal more useful patterns. Accordingly, deep learning models are highly scalable to large input data.

Future work is needed to analyze different types of cancer to identify cancer-specific biomarkers. In addition, there is potential to identify cross-cancer biomarkers through the analysis of aggregated heterogeneous cancer data.

Chapter 5: Dimensionality Reduction of Gene Expression Data for Time-of-death Imputation

5.1 Introduction

Alzheimer's disease is an irreversible neurodegenerative disorder that affects approximately 44 million people worldwide. There is a strong correlation between Alzheimer's disease and the elderly, since 95% of the patients develop symptoms after the age of 65. Analyzing the human post-mortem samples reveals changes in rhythmicity during aging. Further investigations are required to understand the mechanism and extent of these changes. Clinical studies have shown that patients with Alzheimer's disease have a disruption in sleep activity rhythms. However, there is a need for large-scale studies to identify the underlying genes and associated expression patterns in aging and Alzheimer's disease. Since human brain samples can only be collected in a post-mortem manner and at irregular periods, most tissue samples are missing the time of death labels. New advanced data science techniques are needed to impute a time of death for each sample accurately. This critical labeling can be an initial step towards revealing rhythmic patterns that might be correlated with aging and Alzheimer's disease. We used the denoising autoencoder (DAE) model to transform post-mortem gene expression data into a more compact representation. We then used the trained encoded representations to predict the time of death in a binary (day/night) classification task. Our high-performance results ascertain the time of death inference, which can be used to create more labeled samples. Having more large labeled samples opens tremendous opportunities to study the changes in oscillatory gene expression and identify gene correlations with aging and Alzheimer's disease.

5.2 Materials and Methods

5.2.1 Data Collection and Pre-Processing

We collected human brain samples from the largest cohort of time-of-death (TOD) labeled gene expression data [31]. There are 420 overall samples of 210 individuals. However, only

147 individual samples have associated TOD labels (294 total samples).

There is a variety in terms of age and phase shift clear from the distribution of the samples (Figure 5.1 A), which needs to be considered when training a classification model. This implies that a machine learning model needs to be selected that is robust to the differences in the data.

Moreover, the density of sampled time points varies significantly, and the samples are non-uniform in time of death (Figure 5.1 B). Therefore, as an initial step, we made a binary label for each sample: TOD ranges from ZT0 to ZT12 labeled as "day," and TOD ranges from ZT12 to ZT24 labeled as "night." There are 173 samples with TOD set as day, and there are 121 samples with TOD set as night. We further split the data into train and test for the binary classification task. The test contains 10% of the data, and the rest was considered as training.

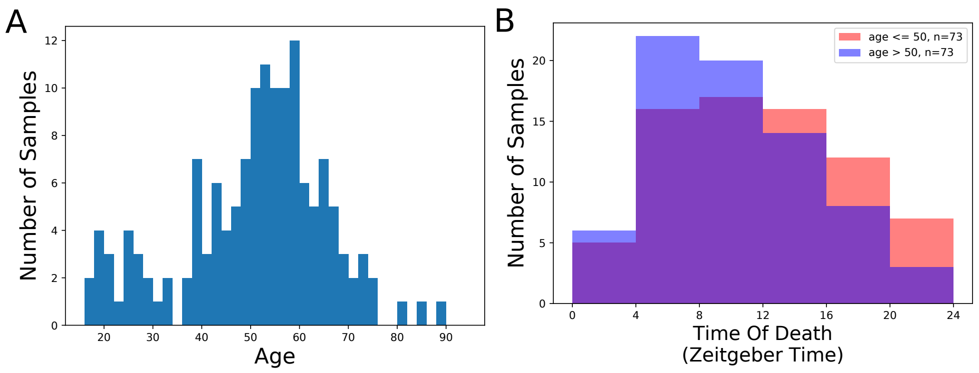


Figure 5.1: This figure represents the Meta-data analysis of the gene expression data and time-of-death data distribution.

5.2.2 Dimensionality Reduction using Denoising Auto Encoder

We used autoencoder (AE) [13] o reduce the dimension of the gene expression data. This step is essential for the TOD classification task since there is a dimensionality curse associated with the gene expression data. In other words, the number of genes (features) is more than the number of samples; this makes the data challenging for the classification task. AE can reduce the dimension of the data while preserving the vital information by reconstructing the input data in the training phase. AE captures the non-linearity information while transforming data into lower-dimensional representation. The main goal of the AE model is to learn reduced dimension features and still be able to reconstruct the input with the low re-construction error. In a high-performance AE model, the encoded data is very similar to the original sample data.

Moreover, to overcome the intrinsic age-bias of the data, we used a denoising strategy [172]. By introducing noise to the input data, the learned AE is capable of capturing meaningful patterns and ignoring any biases associated with the samples.

As an unsupervised pre-training step, we incorporated all of the samples (labeled/unlabeled) to learn more accurate encoded representation. We also generated additional augmented data for pre-training purposes using synthetic minority over-sampling technique (SMOTE) [30]. We further performed supervised fine-tuning using the labeled data. The final encoded representation was extracted for binary classification of the TOD of the human brain samples.

5.2.3 Time-Of-Death Classification using Low Dimensional Representation

As discussed, we inferred TOD as a binary classification. In order to make the most out of the available samples, we performed a leave-one-out (LOO) [81] assessment for the classification task. A support vector machine (SVM) [37] with radial basis kernel function (RBF) [29] was applied for the night/day binary classification using the learned lower-dimensional representations of DAE.

5.3 Results and Discussion

5.3.1 Dimensionality Reduction and Denosing Representation

We trained an unsupervised denoising autoencoder (DAE) model to extract a lower representation of the gene expression data of the human brain. The ideal lower-dimensional features are the set that causes minimum re-construction error in the DAE model architecture. We selected the lower representation of size 300 after tuning this parameter with different values (100, 200, 300, and 500).

Moreover, having larger data size along with synthetic samples, helped our network to have a lower re-constructing error and better generalization.

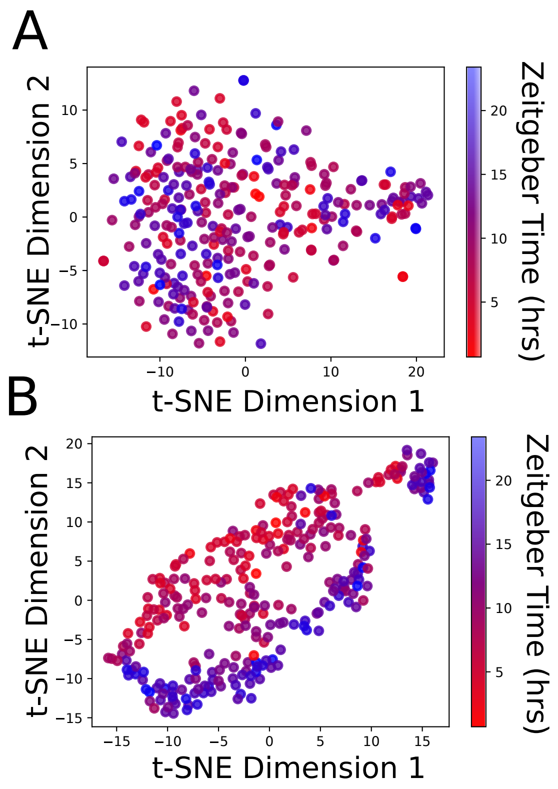


Figure 5.2: t-SNE plots are visualizing gene expression data. **A.** The t-SNE representation of the unprocessed gene expression data, color-coded by TOD. **B.** The t-SNE plot for the reduced-dimensional of the gene expression data, which shows greater separation when color-coded by TOD.

5.3.2 Time-Of-Death Classification Results

We performed a binary classification on the samples to impute TOD (day/night) using the kernel SVM machine learning model. We were able to achieve 95.5% accuracy using the LOO cross-validation approach. The high performance of the classification affirms that DAE was able to encode meaningful features of the data that could benefit the TOD inference.

Widespread phase changes exist during aging in flies' studies [85]. Therefore, we anticipate that the age-dependent classification model could be useful for TOD inference. We tried two additional age-specific classification tasks. First, we trained the SVM model on old samples (greater than 35 years old) and tested on the younger samples. This number is selected based on the age gap in the distribution of the data (Figure 5.1 A). The total samples of 44 were used as a test set. Next, we used young samples for training and old samples for tests. The samples greater than 80 years old were considered as old in this task based on the data distribution. There are only four samples in this category that the SVM model can correctly classify them. The result of age-specific TOP classification is shown in Table 5.1.

The results affirm further validation that our encoded features are robust to noise and the age-bias of the samples.

Table 5.1: TOD classification results using two approaches: **1)** train on the old samples and test on the young population. **2)** train on the young samples and test on the old population.

Test Set	**Accuracy**	**Sensitivity**	**Specificity**	**Precision**	**F1-Score**
Young Samples	95.45	93.75	100.00	100.00	96.77
Old Samples	100.00	100.00	100.00	100.00	100.00

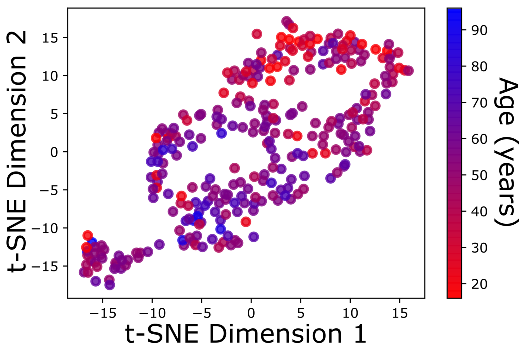


Figure 5.3: t-SNE plot of DAE-features, color-coded by the age of the individuals reveals the age stratification.

5.3.3 Interpretation of Reduced Representation

To interpret the learned feature set and to further confirm the validity of the DAE technique, we compared the feature set of the original data with that of the newly generated data in 2-dimension t-SNE representation [102] (Figure 5.2). This comparison depicts the strength of the DAE method and also assists us to observe the distinction between TODs.

The t-SNE representation of the encoded features also explains the high performance of the classification task. There are some areas in the plot that are not linearly separable; however, the RBF kernel could benefit when transforming the samples to a higher dimension space.

Besides, we observed some amount of age stratification when the age-based t-SNPE representation was used. We hypothesize that there exist considerable differences for different ages that may correspond to the up-regulated genes.

5.4 Summary and Conclusion

The inference of TOD in gene expression data opens an excellent possibility to investigate the pathways and functional annotations associated with aging and Alzheimer's disease. This step is critical as the existing samples are untimed since they are sampled at random time points.

We used DAE methodology along with the SVM model to first extract the compact representation of the post-mortem human brain gene expression samples and then infer the day/night of death through a binary classification task. We further interpreted the learned representation of the DAE by comparing it to the original samples using t-SNE plots. The comparison showed that, in fact, lower-dimensional representations, learned by the autoencoder, could separate the day/night of death despite the noise and age-biases of the data affirming the robustness and generalization of the DAE model. We also observed that the new compact features have some age-stratification that may be because there exist significant variations for different ages. This step needs to be further investigated to figure out potential connections to the up-regulated genes.

Future work is required to identify the genes that played an important role and assisted autoencoder to drive the lower compact representations. This interpretability step is possible by analyzing the weights of the autoencoder model. The genes identified in this step can be used to analyze the biological function by investigating their statistical enrichment of Gene Ontology (GO) terms.

Chapter 6: Interpretation of the Rules Learned by a Deep Convolutional Neural Network to Recognize Core Promoters

6.1 Introduction

During transcription, DNA is copied into RNA by RNA polymerase and a multitude of transcription factors, collectively known as the transcription initiation complex. Transcription factors guide RNA polymerase to the site of transcription initiation by binding to key motifs in the DNA sequence, including the core promoter. Core promoter elements, which are sequence features located within the core promoter, contribute to the regulation of transcription initiation. Core promoters include the $5'$ end of genes and contain transcription start sites (TSS). Typically, these elements are binding sites that are recognized by the molecular complex transcription factor IID (TFIID). Formation of the transcription initiation complex is essential for development and survival in eukaryotic organisms [24].

Core promoters have been studied computationally using position-specific scoring matrices (PSSMs) [73] and sequence similarities comparison [182], as well as traditional machine learning techniques such as decision tree [42], support vector machines (SVM) [11], hidden markov model (HMM) [119], and nearest neighbor search [53] with pre-defined promoter-specific features. While these methods have been successful for identifying several core promoter elements and annotating TSSs, several challenges remain. First, integrative databases of human promoter sequences compute the frequency of occurrence of the TATA box to be around 10 % [183], while other studies vary considerably between 11 % [80] and 17 % [72]. These studies are lower than expected given the prominent role of the TATA box in models of core promoters, and the variability between estimates suggests new approaches are needed to count their occurrence. Furthermore, while studies have shown that deep CNNs are capable of accurately identifying TSS locations [152, 167, 128, 168], have not been characterized. The learned weights could provide insight into which sequence patterns are used by the network to locate TSS positions and to improve characterization of the motifs. The remaining challenge is to develop methods to better understand the reasoning behind the decision-making process.

The challenge of model interpretability demands intelligent systems that can both uncover new core promoter motifs and provide insight into the process of identifying core promoter motifs. In this paper, we present DeepTSS, a deep CNN capable of distinguishing core promoter sequences from other random genomic regions, while focusing on two main goals: improving training strategies and model interpretation. We improved training strategies by addressing class-imbalance due to non-TSS regions greatly outnumbering TSS regions, and used adversarial strategies to improve the detection of sequence patterns. We investigated beyond what has been done previously to interpret the model by analyzing the deeper layers of the network to observe their relationships and discover new meaningful patterns.

6.2 Materials and Methods

The goal of this work is to improve performance and interpetability of deep learning approaches for core promoter detection. In order to achieve high performance of the model, we need to collect and process the appropriate data for training and validating the model. Moreover, we utilize various training strategies to improve model generalizability and to detect meaningful and complex biological patterns. Finally, we analyze filters and weights between convolutional layers to discover important factors leading to model decision-making.

6.2.1 Classification Task

We performed binary classification of DNA sequences of length 251 bp with a defined span of -200 to +50 relative to the TSS. Our trained model performs this classification task exclusively from input DNA sequences without incorporating pre-defined features. We converted each DNA sequence to a one-hot encoded representation, allowing for visualization of DNA sequence patterns in a 2D image-like format (Figure 6.1). A deep CNN classification model identifies patterns through filter activation and integrates them in higher layers of the network. CNNs have a powerful architecture for 2D image classification and object detection [84], and also perform well with analogous 1D sequence classification and motif detection tasks. The architecture of the implemented CNN model is illustrated in Figure 6.1.

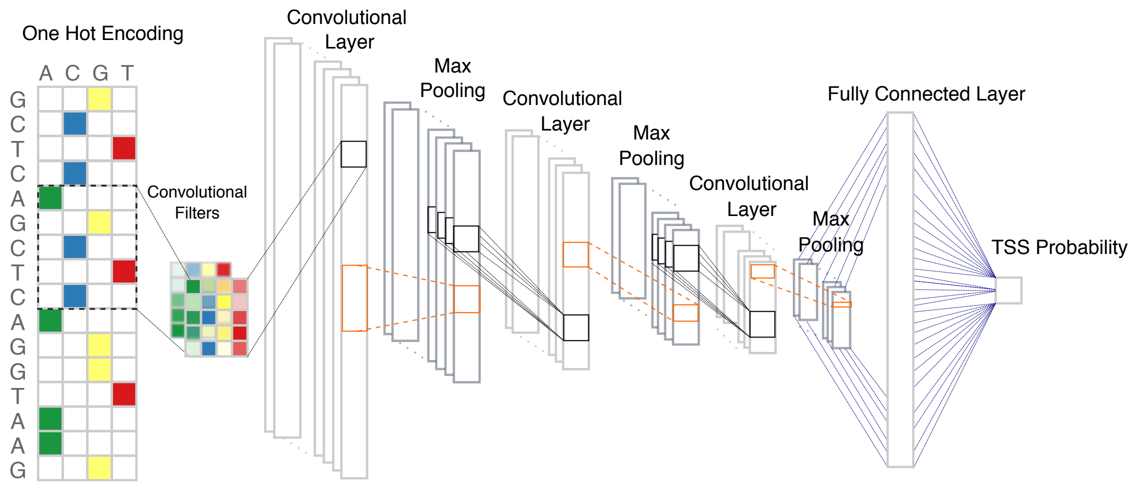


Figure 6.1: Architecture of the convolutional neural network (CNN) model used to train our TSS classification tool. The input sequences are converted into one-hot encoding representation and then are input to the first convolution layer. Our model features three layers of convolution followed by a max pooling layer. The network ends with a fully connected layer and a sigmoid activation function to produce the binary classification prediction.

6.2.2 Data set and data pre-processing

We performed model training and evaluations on the data set from FANTOM5 [117], which includes experimental data from Cap Analysis Gene Expression (CAGE)[159], mapping active promoter regions in mammalian primary cell lines.

Each tag in the CAGE data has an associated enrichment score representing read counts [97]. We filtered the CAGE data based on score in order to collect a higher- confidence set of promoters. We set the threshold to 500 to remove low-coverage TSS regions based on the empirical score distribution (Figure 6.2A). This pre-processing step resulted in 132, 714 promoter sequences. To evaluate the power of our model without chromosome-level bias, we set aside an entire chromosome for testing and performed training and validations on the remaining chromosomes. Using the held-out chromosome allows us to evaluate promoter sequences on this chromosome from other data resources. Previous research demonstrated that this technique is a proper approach to evaluate the model when dealing with cross-chromosome data [143]. We set aside chromosome 21 as a test set similar to [78] and then used a random 10% of the sequences from the remaining chromosomes for validation. The remaining sequences served as a training set for both positive and negative samples. This validation set is used to compare all of the following training steps.

To assess the generalizability of our model, additional tests were performed using two different databases. The eukaryotic promoter database (EPD) [46, 122] is a database containing experimentally-validated promoters. Additionally, gene locations were obtained from Washington University St. Louis epigenome browser (WUSTL)[188]. In order to maintain the consistency of our methodology, we tested the final CNN using chromosome 21 from each of these resources.

6.2.3 Training Strategies

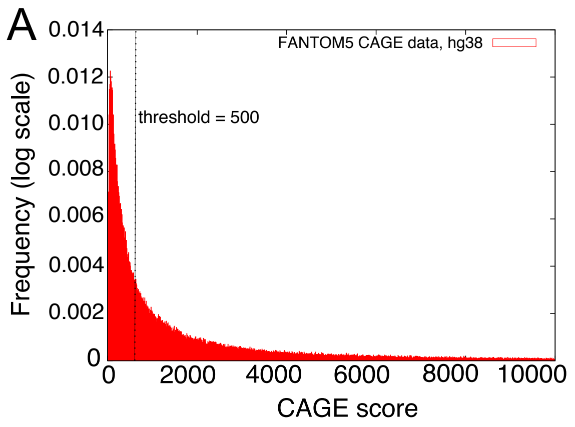
Our training strategies accomplished two main goals. First, we improved the discovery and identification of meaningful biological sequence patterns in promoter sequences. Improved motif detection could help biologists understand the sequence-specific rules that direct the transcriptional machinery. Second, we addressed the large discrepancy between the number of genomic regions overlapping a TSS compared to those without a TSS.

6.2.3.1 Using random genomics sequences as negative samples

There are many more non-promoter regions than promoter regions in the human genome. In order to have a negative set to train the CNN model, we randomly selected non-promoter sequences proportionally to our promoter sequences. We also sampled negative sequences from each chromosomes in proportion to the number of positive promoter sequences to attain a more-balanced data set for training and testing. We then set aside 10 % of the data for validation.

6.2.3.2 Using k-shuffled adversarial data as negative samples

Because of the diverse composition of non-promoter sequences, sampling a small set of non-promoters for training results in weaker generalization over unseen negative data. We addressed imbalance in the negative set by generating adversarial negative samples with the k-shuffling algorithm [71] while maintaining the number of each k-mer in the positive set. The higher the k, the more similar the negative sample is to the positive one, and therefore more challenging to the CNN network. At the same time, the ubiquity of simple



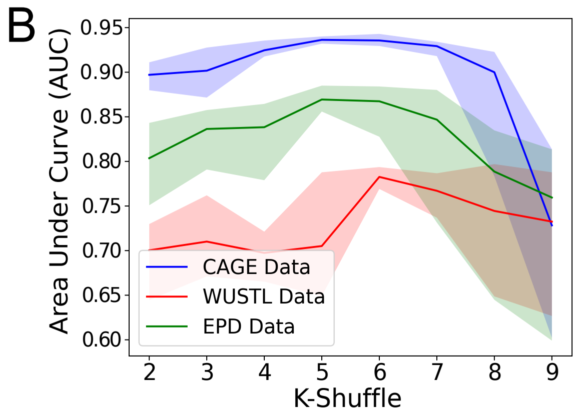


Figure 6.2: **A.** Distribution of the CAGE tag scores. Larger tag scores indicate higher confidence in the data. We selected a threshold of 500, based on this histogram, to obtain a more refined set for training our convolutional neural network (CNN). **B.** The area under the curve (AUC) of the CNN model for different negative samples (k-shuffled) is shown across three datasets. For all three cases, k=6 is challenging but has the least affect on the prediction performance, which makes it a good candidate for generating adversarial negative samples.

sequence features, including GC-content and dinucleotide frequencies, between positive and adversarial negative sequences forces the model to learn more complex sequence features.

This strategy was first introduced in [35] for CNN-based prediction of enhancer sequences. The authors demonstrated that the new set of negative examples used for pre-training forced the model to pay attention to larger and more complex patterns in sequences rather than short, low complexity motifs. We utilized the same training strategy in identification of TSS-containing sequences. Our goal was to attain better generalization of TSS identification and also to challenge the model to learn complex motifs.

We selected the value for k by comparing prediction performance of models trained on generated adversarial negative samples derived from the three available promoter resources. All three models were validated with the same set to provide consistency in picking the right value for k. The AUC values for tuning the value of k during validation are shown in Figure 6.2B for each data set. We selected the value of $k = 6$, which has the highest performance.

6.2.3.3 Using both k-shuffled sequences and random genomics sequences

An additional training strategy consists of using unbalanced data for training the CNN model. In [78], the authors proposed a new training strategy that included X-fold more negative than positive samples where X was set to intervals from 1 to 100. They showed that this training technique reduced false positives when validated with a held-out test set. However, even though the model detected non-promoter regions accurately, TSS prediction was not demonstrated. A combination of k-shuffled and random genomic sequences in the negative set could help to reduce false positive predictions and improve detection of meaningful learned patterns.

6.2.3.4 Using k-shuffled adversarial data to pre-train the convolutional filters

To improve the performance of the model and improve motif detection during training, we implemented an approach that was similar to one used in the enhancer identification task [35]. First, we trained the CNN model using the k-shuffled adversarial negative samples

($k = 6$) and further used the pre-trained convolutional filters to train a new model using random negative samples. We basically, planted pre-trained CNN filters in the new model in order to achieve a better initialization for the training and hence more accurate prediction. Moreover, this strategy forces the model to learn more significant motifs that further assists in model interpretation.

The comparison of information content and sequence complexity of filters in CNN with different training strategies are shown in Figure 6.3. This analogy validates the fact that negative adversarial k-shuffled samples assist the model to learn more complex motifs (filters).

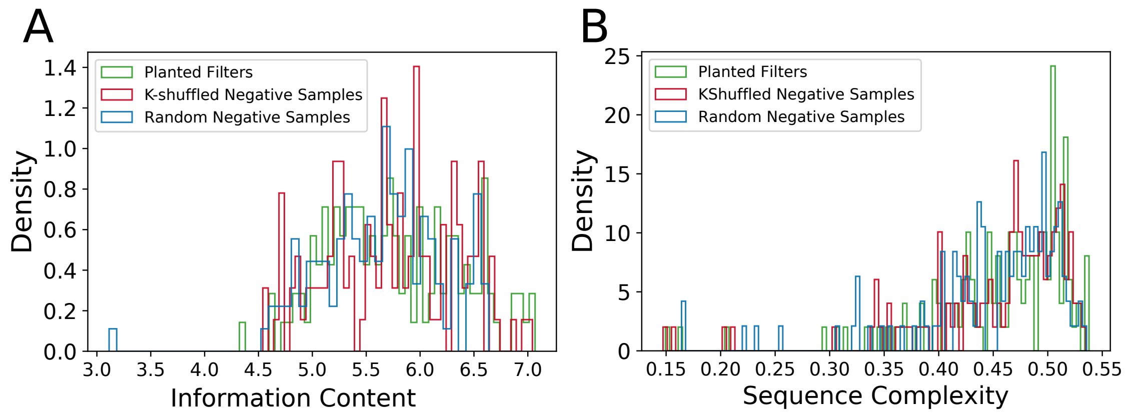


Figure 6.3: Comparison of Information content and sequence complexity across different training strategies.

6.2.3.5 Model Parameter Tuning

we tuned the parameters for the neural network in order to improve classification performance using Hyperas [125], a keras adapted wrapper around Hyperopt [17], that utilizes Baysian hyperparameter search [151]. We sampled over a range of hyperparameter values for the convolution kernel size, number of kernels, pooling map size, hidden units, and dropout rates. We applied hyperparameter search on 10 different runs and selected the top 3 models with the highest overall mean. To further choose the best model, we performed an additional 10 training replicates on the top 3 hyperparameter combinations. Figure D.1 and D.2 illustrate the AUC of these replicates over the different models.

6.2.3.6 Model Interpretability

There are complex patterns around the TSS that help with the activation of gene transcription. To identify these biologically meaningful elements, we need to dive deep into the convolutional neural network and its learned parameters. The aim of this section is to understand the logic behind the convolutional neural network and essentially discover the learned features that helped with precisely identifying the TSS locations in sequences.

Before moving forward with methods to discover new patterns from the CNN model, we need to define several terms that are necessary to consider in finding motifs.

Positional Weight Matrix. To discover a motif in a subset of sequences, first the frequency of each nucleotide over all the samples is captured in each position, which is simply called position frequency matrix (PFM). A position probability matrix (PPM) is then formed by normalizing the frequencies with total number of sequences

$$M_{b,i} = \frac{1}{N}\sum_{j=1}^{N} I(X_{j,i} = b) \tag{6.1}$$

where N is the total number of sequences; $i \in 1, 2, ..., k$, b refers to each base pair (A,C,G, or T); and I is an indicator function. Later, the elements of PWM are calculated as log likelihood of the PPM, which is a general representation of a motif. The PWMs can also be graphically represented using WebLogo tool [39]. In a CNN network, each convolutional filter of the initial layer is believed to be a positional weight matrix (PWM) [157] that are optimized as part of the training to identify relevant patterns in sequences.

Sequence Complexity. Deep learning models can learn complex structures and patterns from the data itself. Motif complexity can be considered to evaluate the extracted patterns from the trained network. The complexity of a motif sequence using the Wootton-Federhen calculation [178] is defined as follows:

$$C_{WF} = \frac{1}{k}\log_4(\frac{k!}{n_A!n_C!n_G!n_T!}) \tag{6.2}$$

where k is the number of nucleotides in the sequence.

Motif Information Content. The information content of a motif explains how well the motif is defined compare to the probability of each nucleotide in the whole genome, which is referred to as background model. The information content is calculated as follows:

$$IC = -\sum_{b,i} p_b \log_2(p_b) + \sum_{b,i} f_{bi} \log_2(f_{ib}) \tag{6.3}$$

where p_b refers to the background model.

Changes in Score. This value represents the variation of the performance when changing specific weights or an input sequence. Here, the sum squared error calculation is used as the change in score:

$$SSE = \sum_{i}^{n} (S_{model} - S_{init})^2 \tag{6.4}$$

where n is the total number of sequences we consider in this calculation and S refers to the score of the model before going into the sigmoid function.

The main purpose of our model interpretation task is to extract and evaluate a set of motifs from the CNN network that potentially have high complexity, information content, and changes in score, which leads us to validate the accountability of the model with an existing motifs and plausibly identify new biologically meaningful motifs.

We further utilized a model interpretability technique that identifies the patterns learned by the model in a backward manner. Using this technique, we can observe the effect of each layer on the input or any other intermediate layers through a method called saliency map [148]. The basic idea behind saliency map is to assume that the prediction model S is differentiable with input $x \in \mathbb{R}^{4\times\ell}$. The variable x refers to a one-hot encoding DNA sequence of length ℓ. Consider the Taylor series expansion of the score $S(x)$ around the term x_{bi} of the matrix x,

$$\begin{aligned} S(x) &= S(x^{(0)}) + \left.\frac{\partial S}{\partial x_{bi}}\right|_{x_{bi}^{(0)}} (x_{bi} - x_{bi}^{(0)}) + ... \\ &= S(x^{(0)}) + \left.\frac{\partial S}{\partial x_{bi}}\right|_{x_{bi}^{(0)}} x_{bi} - \left.\frac{\partial S}{\partial x_{bi}}\right|_{x_{bi}^{(0)}} x_{bi}^{(0)} + ... \end{aligned} \tag{6.5}$$

Due to the non-linearity of the deep learning models, we cannot directly estimate the influence of each nucleotide in the input sequence on the S. However, for each point in x we can approximate the $S(x)$ around a sample sequence $x^{(0)}$ using the first-order Taylor expansion:

$$S(x) \approx \left.\frac{\partial S}{\partial x_{bi}}\right|_{x_{bi}^{(0)}} x_{bi} + c \tag{6.6}$$

where $c = S(x^{(0)}) - \left.\frac{\partial S}{\partial x_{bi}}\right|_{x_{bi}^{(0)}} x_{bi}^{(0)}$ is a constant. The saliency map, μ is then calculated as the gradient of $S(x)$ with respect to variables of the input sequence x at the point $x^{(0)}$

$$\mu_{bi} = \left.\frac{\partial S}{\partial x_{bi}}\right|_{x_{bi}^{(0)}} \tag{6.7}$$

The terms of this matrix μ_{bi} explain the behaviour of the model relative to a particular base b at position i of the input sequence. Larger gradient values determine the area of input that has highest impact on $S(x)$ if changed, therefore more significant.

The results of our model interpretability will be discussed in details in the next section.

6.3 Results and Discussion

We trained multiple models using different training strategies, performed validation using our unique validation set, and tested on the held-out human chromosome 21.

6.3.1 Results on Validation Set

In order to compare our results from different training strategies, we set aside 10% of both positive and negative samples as a validation set. The negative samples were taken from random genomic regions. The results are shown in table 6.1.

6.3.2 Results on Test Set

To further validate our model performance, we used a held-out set with TSS-containing sequences and random genomic sequences from chromosome 21. By setting aside one chromosome as a test set, we were able to validate the generalizability of our model at the genome-wide level. The results of testing on chromosome 21 are shown in table 6.2.

Table 6.1: Model performance on the validation set using CAGE data. This table depicts the average validation performance over 10 different runs using different training strategies.

Training Strategy	**Accuracy**	**Sensitivity**	**Specificity**	**Precision**	**AUC**	**F1-Score**
Random Negative	92.71	92.34	93.07	93.03	97.89	92.68
K-Shuffled Negative	85.61	90.03	81.19	82.74	93.54	86.22
K-Shuffled + Random Negative	89.99	88.17	91.82	91.52	96.37	89.81
K-Shuffled Planted Filters	**93.02**	**92.59**	**93.44**	**93.39**	**97.99**	**92.99**

Table 6.2: Model performance on held-out test set from chromosome 21 over three data sources. This table demonstrates the performance of the CNN model using human chromosome 21 in CAGE, EPD, and WUSTL datasets.

Test Set	**Accuracy**	**Sensitivity**	**Specificity**	**Precision**	**AUC**	**F1-Score**
CAGE	**91.93**	**92.14**	91.71	91.74	**97.63**	**91.94**
EPD	91.56	89.29	**93.83**	**93.54**	97.02	91.36
WUSTL	86.81	90.08	83.54	84.55	93.44	87.23

6.3.3 Biological interpretation

Two different approaches are proposed for interpreting the deep CNN model: (I) investigate and better understand the impact of each layer on the output. We are also interested in understanding how different filters act together to make an impact on the prediction; (II) perform a backward pass analysis, which involves taking the gradient of the prediction score with respect to the original sequence. This provides information about how the output value changes with respect to a small change in the input, and represents the important region of the sequence that contributed to accurate prediction. Additional details about each proposed method are provided further in the following sub-sections.

6.3.3.1 Model Interpretability Using Forward Pass

The initial step towards model interpretability in our CNN model involved analysing the filters and the weights of the initial convolutional layer. We employed different techniques to extract the patterns learned by the network and to understand the correlation between filter pairs.

Filter Analysis. The weights of the initial convolutional layer correspond to position weight matrices. We took a similar empirical approach as presented in [129] to identify the motifs associated with each filter in the CNN. We first found the activation of each filter by scanning over all positive samples in each position and then filtering out the ones that had positive maximum activation at each position. For each filter, all positions with maximum activation were aggregated. This method is stringent and may exclude less frequently occurring motifs since it only considers maximum activation and discards the rest. The purpose of this step is to understand the enrichment of motifs in TSS-containing samples, and to aid in motif discovery. We accumulated the chunk of sequences starting at the maximum position up to the size of the filter and built a position frequency matrix (PFM), representing it visually as a motif logo.

Pairwise Filter Analysis. We also analyzed the maximum activation in a pair of filters in the initial layer of CNN. As shown in figures 6.4 A and B, we identified pairs of filters within a close distance that together contributed to a longer, meaningful initiator and TATA motif. The initiator motif captured by this analysis is almost identical to the motif consensus found in [66] (BBCA$_{+1}$BW).

Filters Influence. To analyze the influence of each filter in the CNN model, we adopted a similar approach to [75], where we nullified each filter individually and calculated the sum of squared changes in the output score of the model. As part of the same analysis, we captured the information for each filter using equation 6.3. We also color-coded each filter by the degree of its complexity using the Wootton-Federhen calculation 6.2. Our analysis in human CAGE data is illustrated in Figure 6.5.

This analysis led to several different observations. First, the highly influential filters (motifs) have higher GC content. This observation indicates that promoter sequences in

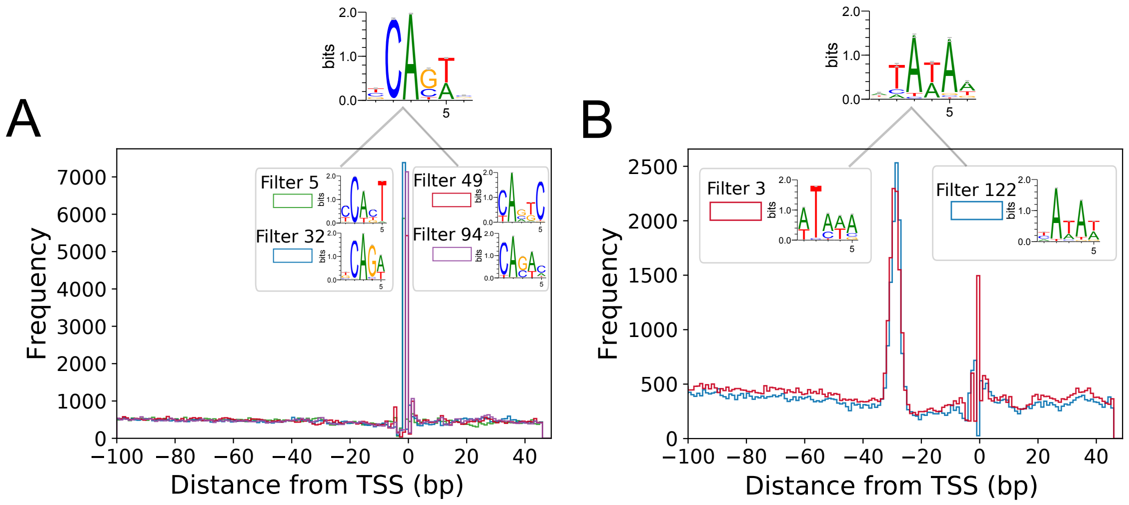


Figure 6.4: Pairwise analysis of filters in the initial layer of the CNN model. We identified pairs of filters that together generated a longer and more complex biological motif. **A.** By looking at the position histogram for every pair of filters in the first later, we discovered four filters that contributed to an initiator motif. **B.** Similarly, paired TATA-containing filters were identified using pairwise analysis technique.

general are GC rich [2] and are therefore more influential on the network when nullified. Second, the initiator motifs have less influence over the model when nullified but have higher information content and complexity. This is due to the fact that there are various initiator-like filters in the CNN that may not be influential individually but in combination have higher impact. We also identified TATA-like filters that have low influence on the model performance when nullified.

Filter Cross-Correlations. We calculated the cross-correlation between every pair of filters in the first layer to measure their similarities and understand their relationships in the network. We first looked at the overall cross-correlation values over all pairs of filters using all CAGE core promoter sequences. Meaning, we only considered the distance between the correlated pairs and ignored their position. We calculated the normalized cross-correlation values considering distances in range $[-5, 5]$ between each pair. This approach helped us to identify highly correlated filters while ignoring their positional biases. We further selected highly correlated filters based on a heuristic threshold of 0.93 and illustrated their linkage

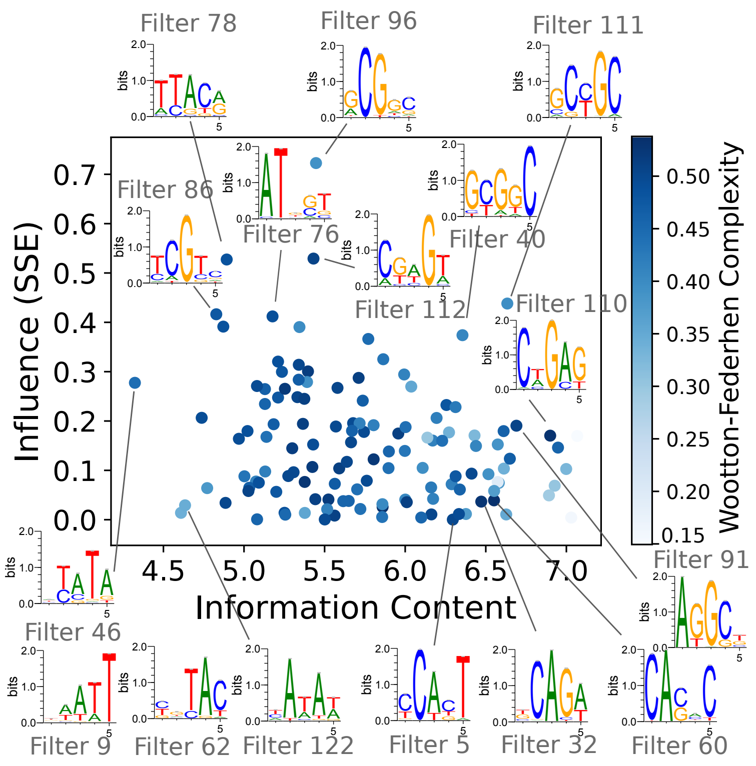


Figure 6.5: Influence of each filter in the initial convolutional layer is captured in human promoter sequences. The scatter plot of information content versus the influence (sum of squares error (SSE)) for each filter in the CNN model was further color-coded with Wootton-Federhen complexity value to capture the sequence complexity. We specifically investigated motifs with high information content and high complexity for which nullification affected the prediction model.

using directed graphs (Figure 6.6). The distance separating filters is also shown in the graph representation. We made several observations while analyzing the linkage graphs:

- The large linkage network on the left of the figure 6.6 depicts the highly correlated GC-containing filters and it is most likely due to the fact that the core promoter sequences are GC-rich by nature. The network also expands to a more T-rich region as we follow the graph to the right. We hypothesis that the filters in this network co-activate to create a larger pattern and possibly an SP1 binding site since it is an enriched motif in human core promoters [183].

- We also observed that the initiator filters 32 and 94 are highly correlated. We previ-

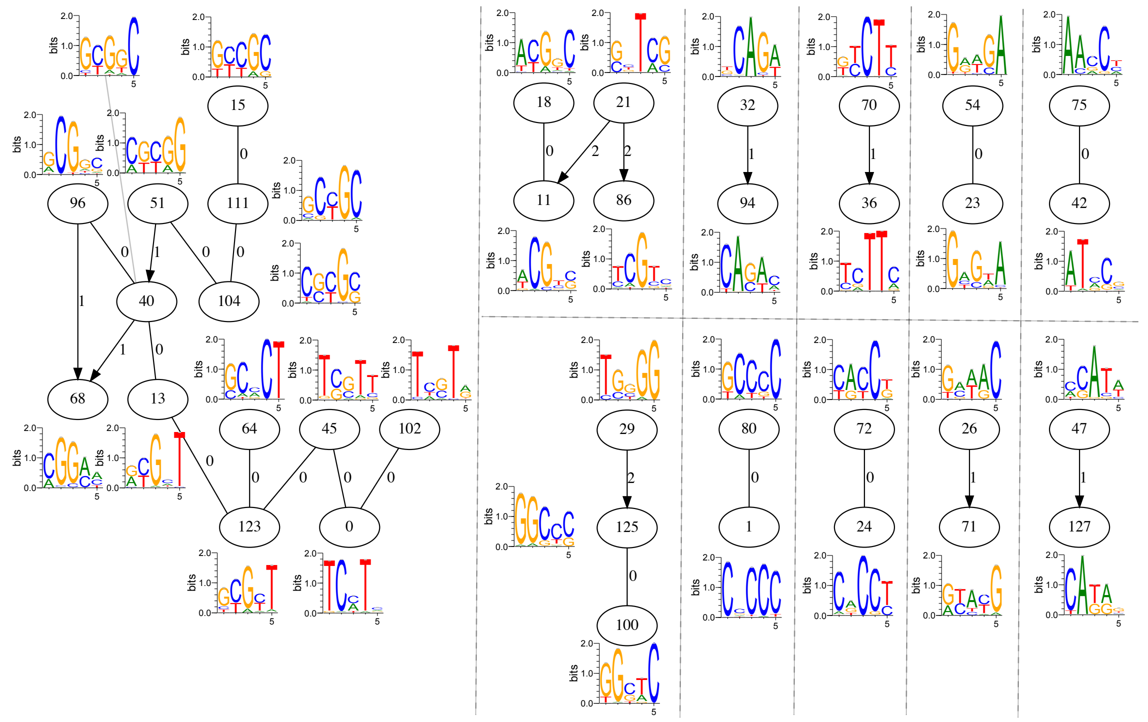


Figure 6.6: The linkage of highly correlated filters of the first convolutional layer are depicted as directed graphs.

ously identified these two initiator-like filters in the pairwise analysis study (Figure 6.4 A).

- We also analyzed the positional frequencies of filters 70 and 36 in the network. We observed that the positions of these filters are enriched near the TSS and together are part of the TCT motif's core (YCTTTY) in human promoters (Figure 6.7 A) [120].
- Another observation is the C-rich pattern of filters 80 and 1. Looking at their position histogram, we discovered that these filters are enriched upstream of the TSS. To the best of our knowledge, there is no study illustrating such a pattern in core promoter sequences. We hypothesize that the CNN learned a novel pattern based on the core promoter sequences, which warrants further analysis and validation.

We also calculated the position-based cross-correlation between pairs of filters. Similar to the previous calculation, we considered the range of distances $[-5, 5]$ between pairs

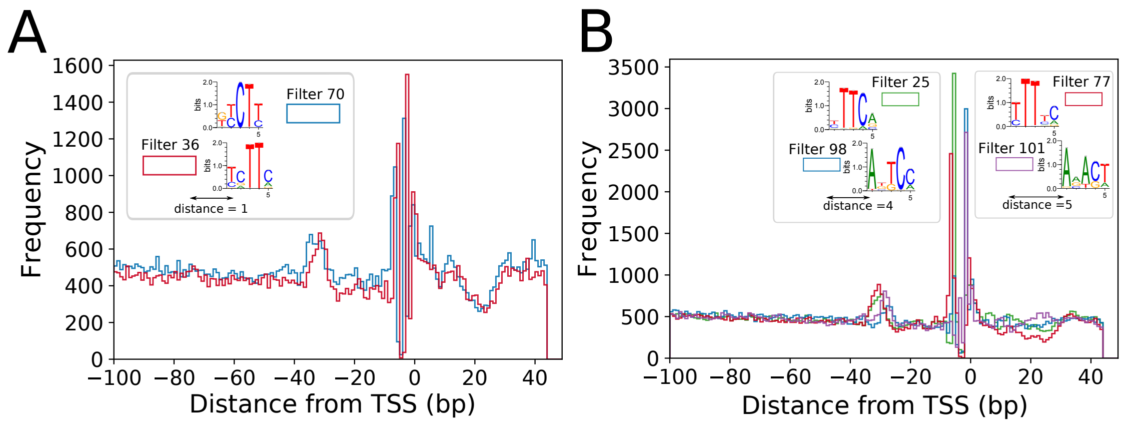


Figure 6.7: Motif-finding through filter correlation analysis. **A.** A TCT initiator motif was identified while investigating the linkage of highly correlated filters. **B.** We identified a T-rich region upstream of the initiators by analyzing the pairwise correlation of the filters.

of filters while also taking their sequence position into the account. We found a set of initiator-like filters that were highly correlated near the TSS. Supporting table D.1 represents the top 10 highly correlated initiator-like filters. Similarly, we detected TATA-like filters while examining highly correlated filters located 25 to 35 bp upstream of the TSS (Supporting Table D.2). Among the identified TATA filters, filter three had the highest positional bias in the TATA-containing region and showed a strong signal for being a TATA motif. This strong TATA filter was activated in 8.65% sequences of CAGE core promoter data in the 25 to 35 bp upstream of the TSS. We also performed the same calculation using the EPD core promoter sequences and identified around 10.99% of sequences with this pattern. This observation is in accord with previous research showing that 10% of sequences contain the canonical TATA box [183]. Moreover, other studies discovered that around 24% of sequences in human core promoters have TATA-like patterns that cannot be unambiguously identified due to their diversity. However, with the CNN model and proper interpretation approaches, we identified TATA-like filters that allowed for the discovery of TATA-like biological patterns and sequences. We concluded that 26.73% and 28.70% of unique sequences from CAGE and EPD core promoters were activated by our TATA filters in table D.2 around the TATA region (25 to 35 upstream of the TSS).

Additionally, we identified a novel, T-rich pattern upstream of the TSS. Figure 6.7B depicts sets of filters that together show a strong T-rich region upstream of the initiator. This pattern was previously identified in yeast core promoter sequences [99]. Future investigation and biological experiments are needed to validate this novel pattern in human core promoter sequences.

6.3.4 Model Interpretability Using Backward Pass

As a second step toward model interpretability we applied a saliency map technique. There are several research papers about genomics-related problems that used saliency maps to understand the relationship between the output of the model with respect to the input [87, 82, 167, 78]. However, to the best of our knowledge no research has been published on identifying core promoter motifs using saliency maps. Our application of the saliency map technique is explained in more detail below:

Impact of Output Layer to Input Sequence. The saliency map assisted in discovering the affect of the model prediction score on each input sequence and identifying the important regions. We calculated the saliency maps from the output layer to each TSS-containing sequence in the CAGE data-set to observe the area that the CNN model mostly focused on. The average saliency values over all sequences in the set is captured in Figure 6.8. There is a strong enrichment around the initiator, which demonstrates that CNN learned the exact TSS region. The maximum average saliency values is right at the TSS location which depicts an enrichment for A nucleotide. The TATA region is noticeable around 25 to 35 bp upstream of the initiator where shows that C and G are depleted in that region (blue color regions in TATA-region). Another observation is the depletion of A and T right after the downstream of TATA-box which deserves a further analysis. There are also some strong patterns downstream of the initiator which requires further evaluation. We applied the same analysis on different sets of EPD sequences: one set only contains initiators and another set that contains only TATA box (Figure D.3 A and B). Similar to CAGE data set, there is a strong enrichment near the transcription start site. However, there is also a strong enrichment in the TATA-box region. This analysis further validate that saliency technique can decipher important patterns that the CNN model learned during the training to detect core promoters.

Impact of Intermediate Convolutional Layers to Input Layer. We also calculated the saliency maps of deeper layers with respect to the input data. The feasibility and application of this approach is depicted in a natural language task [56, 57] for model interpretability. Recent publications in genomics, are mainly focused on the relationship between the model score to the input. It is more valuable to study the impact of each

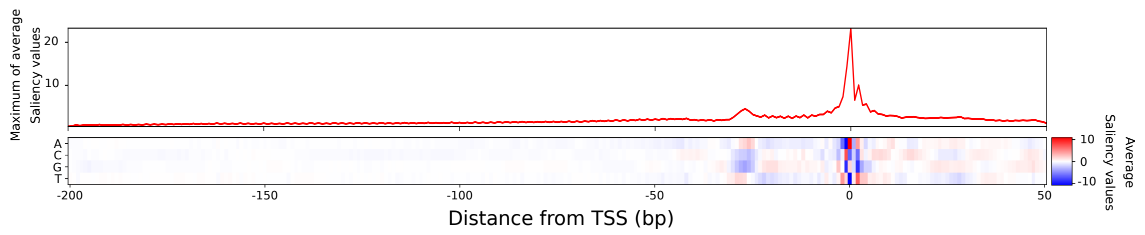


Figure 6.8: Average Saliency maps representation in CAGE data. The region around the initiator shows a strong pattern of initiator motif along with the maximum average absolute values. The TATA region has also strong maps.

layer or more specifically the impact of each convolutional filters on the input. To show the effect of each layer on the input sequence, we selected a TATA-containing promoter from EPD dataset and captured the saliency maps of each layer. As it is shown in Figure 6.9, each convolutional layer contributes toward identifying certain patterns that together direct the network to identify TATA region. We also noticed that the saliency patterns captured from convolutional layer one is very similar to the input sequence, which affirms that it is responsible to capture the simple patterns from the input sequence. However, as presented in the figure the layer 2 triggers TATA region of the sequence. We hypothesize that the second layer is responsible to connect the patterns learned by layer one. As we go deeper in the model, the unimportant regions are depleted and important patterns are highlighted with higher saliency values (either in positive or negative direction). These steps of the saliency vividly depicts that deeper layer of the network are responsible to amplify important aspect of the input. It also represents that the deeper layers are more robust to the variation of the input data.

To further understand the functionality of different layers in our CNN model, we identified motifs with strong enrichment in our promoters set from JASPAR [139] database using fimo [60] algorithm under the meme suite software toolkit [12]. We selected an example (KLF16) motif that showed high enrichment 50 bp upstream of the TSS in promoter sequences (Figure 6.10 A). We then calculated the saliency maps of second layer convolution to these positive sequences. We observed that one filter shows enrichment around 50 bp upstream of the TSS. The extracted gradient values for that filter were normally distributed (data is not shown). We then detected the positions, which had significant gradient values using the false discovery rate (FDR) cutoff of $1e-4$. The frequencies of these significant values were captured with respect to the position of the input sequence

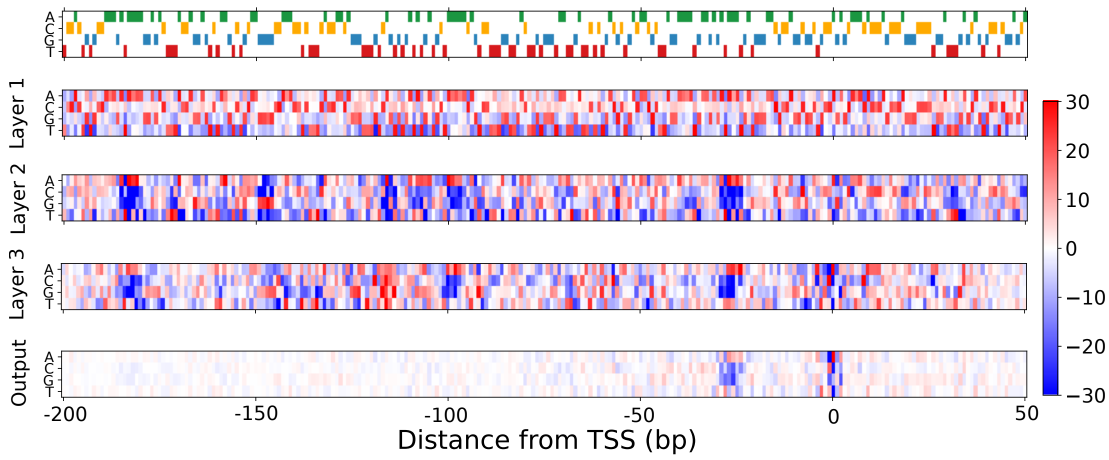


Figure 6.9: Saliency maps on a TATA-containing sequence from EPD database. The effect of each convolutional layer on the input data is captured through saliency map technique. These steps represents deeper layers of CNN are responsible to connect the patters from the initial layers.

(Figure 6.10B). Interestingly, we observed the similar positional trend and enrichment in −50 of the TSS. The result suggest that, we further can incorporate this technique to identify important filters with high enrichment that most likely contribute toward meaningful biological patterns.

Interpreting Mutually dependent Filters in Higher level Convolution Layers. In spite that the first layer of convolution captures simple patterns in the data, the second layer and third layers are responsible to extract the relationship between low level features that captured by the initial layer. In particular, we investigated the connection between the first layer of convolution and the second one. We centred our analysis around the initiator and TATA sequences to perceive the effect of second layer in the CNN model. We calculated the saliency values of the second layer convolution with respect to the lower level filters and discern pairs of filters that contributed most in this map across initiator and TATA-containing samples downloaded from EPD database. The average saliency values along with the maximum positional and filter plots are captured in Figure 6.11 and Figure D.4. Using this method, we could identify more initiator and TATA-like filters that together aid in accurate TSS detection.

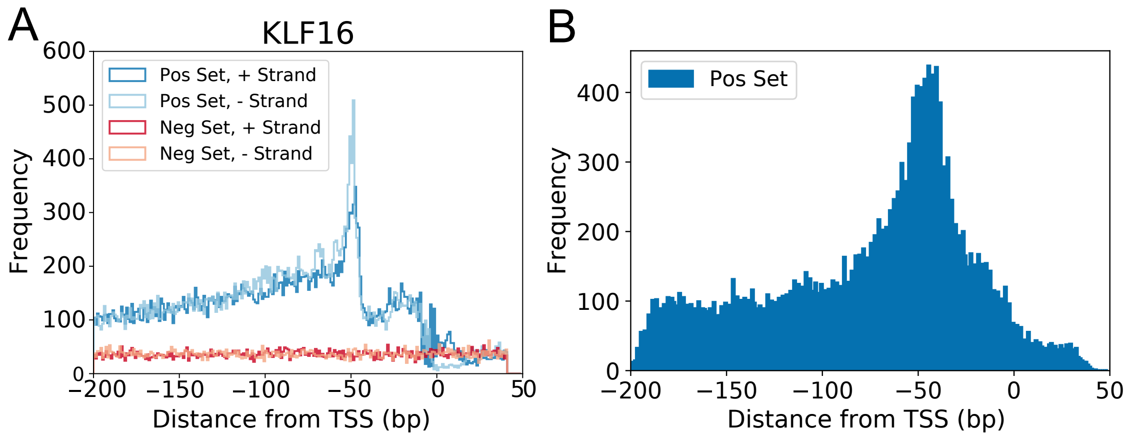


Figure 6.10: Results on understanding the impact of different layer of convolution on the input sequences. **A.** The distribution of a strong motif (KLF16) in the CAGE data. Both positive and negative sequences were captured to represent their difference. **B.** Based on the KLF16 distribution in our positive set, we expected to see the same trend when looking at the most significant gradient values of the convolutional layers with respect to input sequences.

An interesting observation is that some of the previously found initiator filters (94, 49,and 60) were not highly activated in analyzing the EPD initiator sequences, which suggests the initiator motifs in the EPD are not C-rich in the 3 bp downstream of the TSS. Further analysis needs to be done to validate this point.

6.4 Summary and Conclusions

In this work, we presented a high performance CNN model for identifying TSSs in human core promoter sequences. We trained our TSS-detection network using a variety of strategies for the purpose of extracting motifs with high information content and complexity. We challenged the CNN model by incorporating adversarial k-shuffled promoter sequences as a negative set. The k-shuffling strategy maintains the nucleotide composition of the TSS-containing sequences, forcing the CNN model to learn patterns that are more complex. This pre-training step was fine-tuned subsequently using random negative sequences from the human genome. After implementing this training strategy, we built an accurate deep learning model to detect core promoters that we tested on a held-out set from chromosome 21 from three different promoter databases (CAGE, EPD, and WUSTL). Then,

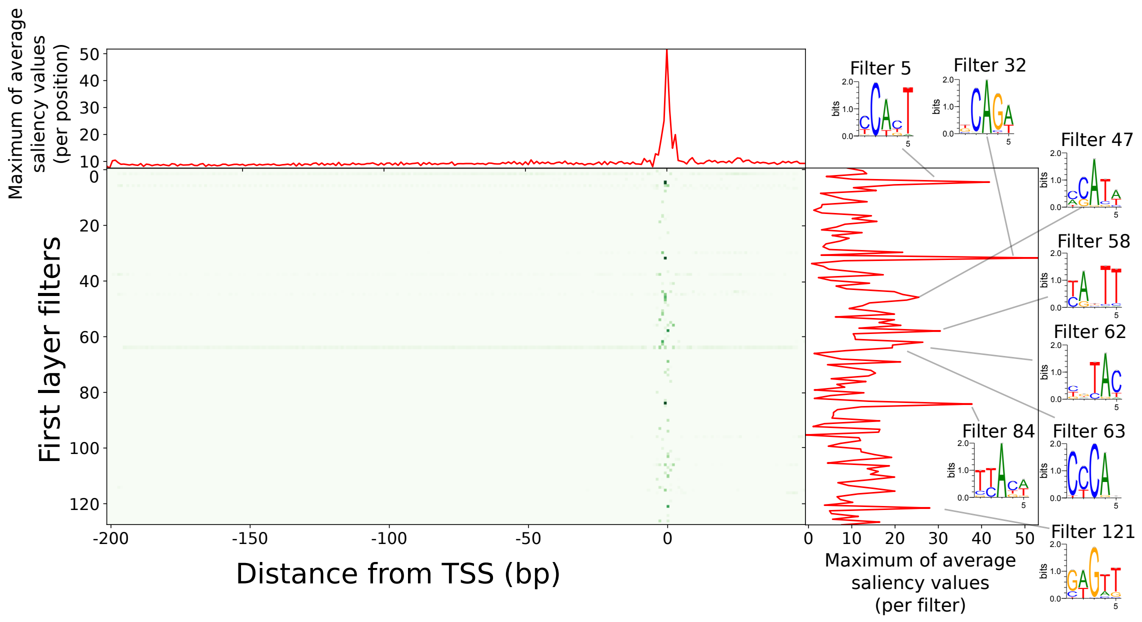


Figure 6.11: The saliency maps of layer 2 convolutional neural network on the first layer convolutional network. This plot represents the effect of second layer on the first layer using the saliency map techniques. The data used in this analysis is initiator sequences from EPD database.

we developed a series of novel interpretability approaches that assisted in deciphering new biological patterns within core promoter sequences and identifying known motifs.

We took two main steps towards network interpretability that can be applied to similar problems:

First, we analyzed the weights and filters learned by the CNN model in a forward manner. We studied the effect of filters learned by the initial layer on the output prediction score. We also identified filters correlated with accurate prediction of initiator and TATA-containing sequences. In addition, we discovered new biological patterns that deserve further computational and *in vivo* analysis. Additional steps include applying the same analysis to other species to determine whether the discovered biological pattern occur in other species as well.

Second, we used saliency maps to examine the influence of the first layer of the CNN model, as well as deeper layers of input. Using this approach, we studied the role of each convolutional layer as well as their effect on the input and the initial layer of the CNN network. We can further study each convolutional layer in the network by summing over the gradients with respect to the input data across different samples. This method provides insight into the functionality of each layer in the complex deep learning model. Another approach involves taking the gradient of each filter in the convolutional layer with respect to the input. This method captures the contribution of each filter towards recognizing core promoter sequences. The goal is then to identify complex meaningful motifs from analyzing the positions of k-mers across samples.

The future direction of this work revolves around interpretability of the deeper layers of the network. Despite developing several novel techniques for explaining the higher layers of the network, there are opportunities remaining for understanding how the network identifies complex patterns of promoter sequences.

Chapter 7: Conclusions

7.1 Summary

In this thesis, we mainly focused on ML and DL methods and their applications in biology and genomics. Our main goal was to introduce novel methodologies to not only solve a specific problem with high performance but broaden the possibilities for discovering new patterns in biological sequences. Advances in ML techniques have introduced tremendous opportunities for deepening our scientific knowledge. However, understanding how a model learns features and patterns is an open area of research. Although model interpretability is an essential part of understanding intelligence systems, it is usually ignored due to the complexity of the model. There is a need to introduce new approaches and methodologies for explaining and interpreting the results in order to gain the trust of end-users of the ML predictive models. Model interpretability can further help us as data scientists to validate ML models, identify and evaluate errors, and take actions to improve.

In this thesis, we looked at model interpretability from three different angles.

1. We first looked at model interpretability as a way to understand the data and its underlying patterns. We developed a method to annotate RNA secondary structures automatically, wherein we could explore patterns and specifications of different RNA structure types. Our annotation approach culminated in the creation of a large, integrated meta-database of RNA secondary structures obtained from seven different data resources, known as bpRNA. Along with secondary structure, bpRNA contains detailed base-pairing information and annotations for each RNA molecule, providing tremendous opportunities for several applications. The dataset generated by bpRNA contains over $100,000$ RNA structures, benefiting the training of ML models on RNA-related problems.

2. We then achieved interpretability as we built a model for detecting pseudoknots in RNA secondary structures. We introduced novel features specific to RNA pseudo-knot structures and trained a rule-based ML model to classify pseudoknot-containing

sequences from other RNA sequences. We captured the importance of features by investigating the rules learned by the model during training.

3. Model interpretability in DL models is more critical than ML models since DL models are highly complex by nature and do not accept pre-defined features. Understanding the decision-making process underlying DL models, including identifying patterns and features learned by the model, is an open area of research. With this in mind, we explored different DL applications and developed novel interpretability techniques specific to each problem.

    - We implemented a stacked denoising autoencoder to reduce the dimensionality of gene expression profiles in breast cancer samples. We then incorporated the compact representations to classify cancerous vs non-cancerous samples. To identify essential genes and interpret what the model learned, we analyzed the weights of the deeply connected layers of the autoencoder. We extracted a set of genes, known as deeply connected genes (DCGs), that we hypothesized the DL model focused on during training. We then tested the DCGs in the cancer classification task to validate their importance. We also examined the functional enrichment of the DCGs through a GO term and Panther pathway analysis. Our novel methodology can be used to identify important genes for cancer prediction and biomarker discovery in any cancer.
    - We used denoising autoencoder methodology to extract encoded features of gene expression samples from post-mortem human brain tissue. Our approach takes an initial step towards time-of-death inference that will open the possibility for large-scale investigation of how daily rhythms of gene expression change with age and Alzheimer's disease. We inferred the time of death through a binary classification task using a kernel support vector machine (SVM). We interpreted the extracted features to test the validity of the proposed approach further. The t-SNE representation of the lower dimensional features clearly showed the capability of this method to impute the time of death. Moreover, we achieved high performance in classifying the time of death when using age-dependent samples for training and testing the model. Our results confirm that our approach is robust to noise and age-related biases in the data.
    - We used a deep CNN to recognize core promoter sequences in DNA and inves-

tigate how the model identified complex biological motifs. Our CNN, in concert with new training strategies, accurately identified TSSs. We developed two different approaches to interpret the weights of the model and to explain how motifs were discovered. Our first approach involved investigating the weights of the CNN from the first convolutional layer to the output. We developed methods to identify initiator and TATA-containing sub-motifs and made additional findings that warrant further investigation. We also studied the correlation between the first and second layers with pairwise synergy analysis. Our second approach involved analyzing the effect of the predicted output on the input using a backward strategy. We also studied the effect of the second layer on the first layer. Through our analysis, we demonstrated how different layers cooperate to predict TSSs. To the best of our knowledge, no previous research has involved model interpretation at such a granular level. Our novel techniques for explaining the rules of the model can be used for similar applications in data science to understand the underlying decision-making of CNNs or other DL models.

The focus of this thesis was developing novel methodologies to discover new patterns in biological sequences. We cover interpretability from understanding patterns of an RNA structure to detecting genes involved in cancer, to inferring time-of-death from gene expression samples of post-mortem human brain, all the way to identifying motifs associated with core promoters in biological sequences.

7.2 Future Directions

The next step in the bpRNA project will be to go beyond the single-molecule and consider all sequences in the database. Further exploratory analysis will be performed for each RNA structure type to better understand the relationship between RNA sequence, structure, and function.

To improve pseudoknot structure prediction, we can use the available data in bpRNA. Future work on this project will involve performing the same analysis with a larger dataset like bpRNA. The next step in this project will be to implement a gradient boosting technique and compare the performance with previous results. Another approach to consider for model interpretability is SHapley Additive exPlanations (SHAP) [100], which is compatible with tree-like ML approaches and can quickly and precisely extract important attributes involved in driving classification prediction.

Further, we can apply SDAE on combinations of cancer types and do pan-cancer analysis to detect essential genes that are involved in various types of cancer. Using the available data from TCGA, we can study the similarities and variations in different types of cancers. We can also use gradient-based attribution methods from [5] to asses important genes in the cancer genomics data. Besides, the more in-depth model interpretability of SDAE is needed to identify genes associated with pathways in aging and Alzheimer's disease.

To better understand the functionality and patterns in core-promoter elements, we can apply the same type of analysis to other species and compare the results across species.

Further interpretability approaches are also possible to study a higher level of the CNN model. For example, an additional step is to apply saliency maps to different layers and understand the effect of each layer. A potential future method to study core promoter sequences involves applying an attention mechanism to the existing CNN [171]. An attention mechanism directs the focus of the model toward some regions of the input, resulting in more accurate performance. This technique could potentially aid in identifying additional, prominent biological motifs.

Bibliography

[1] Peter L Adams, Mary R Stahley, Anne B Kosek, Jimin Wang, and Scott A Strobel. Crystal structure of a self-splicing group i intron with both exons. *Nature*, 430:45–50, 2004.

[2] Pelin Akan and Panos Deloukas. Dna sequence and structural properties as predictors of human and mouse promoters. *Gene*, 410(1):165–176, 2008.

[3] Tatsuya Akutsu. Dynamic programming algorithms for rna secondary structure prediction with pseudoknots. *Discrete Applied Mathematics*, 104:45–62, 2000.

[4] Uri Alon, Naama Barkai, Daniel A Notterman, Kurt Gish, Suzanne Ybarra, Daniel Mack, and Arnold J Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences*, 96(12):6745–6750, 1999.

[5] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. *arXiv preprint arXiv:1711.06104*, 2017.

[6] Ebbe Sloth Andersen, Magnus Alm Rosenblad, Niels Larsen, Jesper Cairo Westergaard, Jody Burks, Iwona K Wower, Jacek Wower, Jan Gorodkin, Tore Samuelsson, and Christian Zwieb. The tmrdb and srpdb resources. *Nucleic acids research*, 34(suppl_1):D163–D168, 2006.

[7] Mirela Andronescu, Vera Bereg, Holger H Hoos, and Anne Condon. Rna strand: the rna secondary structure and statistical analysis database. *BMC bioinformatics*, 9(1):340, 2008.

[8] Mirela Andronescu, Vera Bereg, Holger H Hoos, and Anne Condon. Rna strand: the rna secondary structure and statistical analysis database. *BMC bioinformatics*, 9:1, 2008.

[9] Mirela Andronescu, Anne Condon, Holger H Hoos, David H Mathews, and Kevin P Murphy. Efficient parameter estimation for rna secondary structure prediction. *Bioinformatics*, 23(13):i19–i28, 2007.

[10] Mirela Andronescu, Anne Condon, Holger H Hoos, David H Mathews, and Kevin P Murphy. Computational approaches for rna energy parameter estimation. *RNA*, 16(12):2304–2318, 2010.

[11] Firoz Anwar, Syed Murtuza Baker, Taskeed Jabid, Md Mehedi Hasan, Mohammad Shoyaib, Haseena Khan, and Ray Walshe. Pol ii promoter prediction using characteristic 4-mer motifs: a machine learning approach. *BMC bioinformatics*, 9(1):414, 2008.

[12] Timothy L Bailey, James Johnson, Charles E Grant, and William S Noble. The meme suite. *Nucleic acids research*, 43(W1):W39–W49, 2015.

[13] Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 37–49, 2012.

[14] Stanislav Bellaousov and David H Mathews. Probknot: fast prediction of rna secondary structure including pseudoknots. *Rna*, 16:1870–1880, 2010.

[15] Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007.

[16] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the royal statistical society. Series B (Methodological)*, pages 289–300, 1995.

[17] James Bergstra, Dan Yamins, and David D Cox. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. In *Proceedings of the 12th Python in science conference*, pages 13–20. Citeseer, 2013.

[18] Helen M Berman, Wilma K Olson, David L Beveridge, John Westbrook, Anke Gelbin, Tamas Demeny, Shu-Hsin Hsieh, AR Srinivasan, and Bohdan Schneider. The nucleic acid database. a comprehensive relational database of three-dimensional structures of nucleic acids. *Biophysical journal*, 63(3):751, 1992.

[19] Gérard Biau. Analysis of a random forests model. *The Journal of Machine Learning Research*, 13:1063–1095, 2012.

[20] Christopher M Bishop. Pattern recognition. *Machine Learning*, 2006.

[21] Olivier Bousquet and Léon Bottou. The tradeoffs of large scale learning. In *Advances in neural information processing systems*, pages 161–168, 2008.

[22] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

[23] James W Brown. The ribonuclease p database. *Nucleic acids research*, 26(1):351–352, 1998.

[24] Jennifer EF Butler and James T. Kadonaga. The rna polymerase ii core promoter: a key component in the regulation of gene expression. *Genes development*, 16(20):2583–2592, 2002.

[25] Jamie J Cannone, Sankar Subramanian, Murray N Schnare, James R Collett, Lisa M D'Souza, Yushi Du, Brian Feng, Nan Lin, Lakshmi V Madabusi, Kirsten M Müller, et al. The comparative rna web (crw) site: an online database of comparative sequence and structure information for ribosomal, intron, and other rnas. *BMC bioinformatics*, 3(1):2, 2002.

[26] Jamie H Cate, Anne R Gooding, Elaine Podell, Kaihong Zhou, Barbara L Golden, Craig E Kundrot, Thomas R Cech, and Jennifer A Doudna. Crystal structure of a group i ribozyme domain: principles of rna packing. *Science*, 273(5282):1678–1685, 1996.

[27] Kung-Yao Chang and Ignacio Tinoco. Characterization of a" kissing" hairpin complex derived from the human immunodeficiency virus genome. *Proceedings of the National Academy of Sciences*, 91(18):8705–8709, 1994.

[28] Yin-Wen Chang, Cho-Jui Hsieh, Kai-Wei Chang, Michael Ringgaard, and Chih-Jen Lin. Training and testing low-degree polynomial data mappings via linear svm. *The Journal of Machine Learning Research*, 11:1471–1490, 2010.

[29] Yin-Wen Chang, Cho-Jui Hsieh, Kai-Wei Chang, Michael Ringgaard, and Chih-Jen Lin. Training and testing low-degree polynomial data mappings via linear svm. *Journal of Machine Learning Research*, 11(Apr):1471–1490, 2010.

[30] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[31] Cho-Yi Chen, Ryan W Logan, Tianzhou Ma, David A Lewis, George C Tseng, Etienne Sibille, and Colleen A McClung. Effects of aging on circadian patterns of gene expression in the human prefrontal cortex. *Proceedings of the National Academy of Sciences*, 113(1):206–211, 2016.

[32] Hamidreza Chitsaz and Mohammad Aminisharifabad. Exact learning of rna energy parameters from structure. *Journal of Computational Biology*, 22:463–473, 2015.

[33] François Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

[34] Peter Clote, Stefan Dobrev, Ivan Dotu, Evangelos Kranakis, Danny Krizanc, and Jorge Urrutia. On the page number of rna secondary structures with pseudoknots. *Journal of mathematical biology*, 65(6-7):1337–1357, 2012.

[35] Dikla Cohn, Or Zuk, and Tommy Kaplan. Enhancer identification using transfer and adversarial deep learning of dna sequences. *bioRxiv*, page 264200, 2018.

[36] Carl C Correll, Betty Freeborn, Peter B Moore, and Thomas A Steitz. Metals, motifs, and recognition in the crystal structure of a 5s rrna domain. *Cell*, 91(5):705–712, 1997.

[37] Corinna Cortes and Vladimir Vapnik. Support vector machine. *Machine learning*, 20(3):273–297, 1995.

[38] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20:273–297, 1995.

[39] Gavin E Crooks, Gary Hon, John-Marc Chandonia, and Steven E Brenner. Weblogo: a sequence logo generator. *Genome research*, 14(6):1188–1190, 2004.

[40] Joseph A Cruz and David S Wishart. Applications of machine learning in cancer prediction and prognosis. *Cancer informatics*, 2, 2006.

[41] Kévin Darty, Alain Denise, and Yann Ponty. Varna: Interactive drawing and editing of the rna secondary structure. *Bioinformatics*, 25(15):1974, 2009.

[42] Ramana V Davuluri, Ivo Grosse, and Michael Q Zhang. Computational identification of promoters and first exons in the human genome. *Nature genetics*, 29(4):412, 2001.

[43] Robert M Dirks and Niles A Pierce. A partition function algorithm for nucleic acid secondary structure including pseudoknots. *Journal of computational chemistry*, 24:1664–1677, 2003.

[44] Chuong B Do, Daniel A Woods, and Serafim Batzoglou. Contrafold: Rna secondary structure prediction without physics-based models. *Bioinformatics*, 22:e90–e98, 2006.

[45] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[46] René Dreos, Giovanna Ambrosini, Rouayda Cavin Périer, and Philipp Bucher. Epd and epdnew, high-quality promoter resources in the next-generation sequencing era. *Nucleic acids research*, 41(D1):D157–D164, 2012.

[47] Rasool Fakoor, Faisal Ladhak, Azade Nazi, and Manfred Huber. Using deep learning to enhance cancer diagnosis and classification. In *Proceedings of the ICML Workshop on the Role of Machine Learning in Transforming Healthcare. Atlanta, Georgia: JMLR: W&CP*, 2013.

[48] Matthew A Fountain, Martin J Serra, Thomas R Krugh, and Douglas H Turner. Structural features of a six-nucleotide rna hairpin loop found in ribosomal rna. *Biochemistry*, 35(21):6539–6548, 1996.

[49] Susan M Freier, Ryszard Kierzek, John A Jaeger, Naoki Sugimoto, Marvin H Caruthers, Thomas Neilson, and Douglas H Turner. Improved free-energy parameters for predictions of rna duplex stability. *Proceedings of the National Academy of Sciences*, 83:9373–9377, 1986.

[50] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55:119–139, 1997.

[51] Terrence S Furey, Nello Cristianini, Nigel Duffy, David W Bednarski, Michel Schummer, and David Haussler. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16(10):906–914, 2000.

[52] Hin Hark Gan, Daniela Fera, Julie Zorn, Nahum Shiffeldrim, Michael Tang, Uri Laserson, Namhee Kim, and Tamar Schlick. Rag: Rna-as-graphs database—concepts, analysis, and features. *Nutrition and Health*, 5(1-2):1285–1291, 1987.

[53] Yanglan Gan, Jihong Guan, and Shuigeng Zhou. A pattern-based nearest neighbor search approach for promoter prediction using dna structural profiles. *Bioinformatics*, 25(16):2006–2012, 2009.

[54] David P Gardner, Pengyu Ren, Stuart Ozer, and Robin R Gutell. Statistical potentials for hairpin and internal loops improve the accuracy of the predicted rna structure. *Journal of molecular biology*, 413(2):473–483, 2011.

[55] Paul P Gardner and Robert Giegerich. A comprehensive comparison of comparative rna structure prediction approaches. *BMC bioinformatics*, 5:1, 2004.

[56] Reza Ghaeini, Xiaoli Z Fern, Hamed Shahbazi, and Prasad Tadepalli. Saliency learning: Teaching the model where to pay attention. *arXiv preprint arXiv:1902.08649*, 2019.

[57] Reza Ghaeini, Xiaoli Z Fern, and Prasad Tadepalli. Interpreting recurrent and attention-based neural models: a case study on natural language inference. *arXiv preprint arXiv:1808.03894*, 2018.

[58] Sunny D Gilbert, Robert P Rambo, Daria Van Tyne, and Robert T Batey. Structure of the sam-ii riboswitch bound to s-adenosylmethionine. *Nature structural & molecular biology*, 15:177–182, 2008.

[59] Todd R Golub, Donna K Slonim, Pablo Tamayo, Christine Huard, Michelle Gaasenbeek, Jill P Mesirov, Hilary Coller, Mignon L Loh, James R Downing, Mark A Caligiuri, et al. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *science*, 286(5439):531–537, 1999.

[60] Charles E Grant, Timothy L Bailey, and William Stafford Noble. Fimo: scanning for occurrences of a given motif. *Bioinformatics*, 27(7):1017–1018, 2011.

[61] Aman Gupta, Haohan Wang, and Madhavi Ganapathiraju. Learning structure in gene expression data using deep architectures, with an application to gene clustering. In *Bioinformatics and Biomedicine (BIBM), 2015 IEEE International Conference on*, pages 1328–1335. IEEE, 2015.

[62] ME Harris, AV Kazantsev, JL Chen, and NR Pace. Analysis of the tertiary structure of the ribonuclease p ribozyme-substrate complex by site-specific photoaffinity crosslinking. *Rna*, 3(6):561–576, 1997.

[63] Christian Haslinger and Peter F Stadler. Rna structures with pseudo-knots: Graph-theoretical, combinatorial, and statistical properties. *Bulletin of mathematical biology*, 61(3):437–467, 1999.

[64] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[65] Ivo L Hofacker, Walter Fontana, Peter F Stadler, L Sebastian Bonhoeffer, Manfred Tacker, and Peter Schuster. Fast folding and comparison of rna secondary structures. *Monatshefte für Chemie/Chemical Monthly*, 125:167–188, 1994.

[66] Cassidy Yunjing Huang, Sascha HC Duttke, James T Kadonaga, et al. The human initiator is a distinct and abundant element that is precisely positioned in focused core promoters. *Genes & development*, 31(1):6–11, 2017.

[67] Shengrong Huang, Yun-Xing Wang, and David E Draper. Structure of a hexanucleotide rna hairpin loop conserved in ribosomal rnas. *Journal of molecular biology*, 258(2):308–321, 1996.

[68] M Isobe, BS Emanuel, D Givol, M Oren, and Carlo Maria Croce. Localization of gene for human p53 tumour antigen to band 17p13. 1986.

[69] Matthew K Iyer, Yashar S Niknafs, Rohit Malik, Udit Singhal, Anirban Sahu, Yasuyuki Hosono, Terrence R Barrette, John R Prensner, Joseph R Evans, Shuang Zhao, et al. The landscape of long noncoding rnas in the human transcriptome. *Nature genetics*, 47:199–208, 2015.

[70] Joseph A Izzo, Namhee Kim, Shereef Elmetwaly, and Tamar Schlick. Rag: an update to the rna-as-graphs resource. *Bmc Bioinformatics*, 12(1):219, 2011.

[71] Minghui Jiang, James Anderson, Joel Gillespie, and Martin Mayne. ushuffle: a useful tool for shuffling biological sequences while preserving the k-let counts. *BMC bioinformatics*, 9(1):192, 2008.

[72] Victor X Jin, Gregory AC Singer, Francisco J Agosto-Pérez, Sandya Liyanarachchi, and Ramana V Davuluri. Genome-wide analysis of core promoter elements from conserved human and mouse orthologous pairs. *BMC bioinformatics*, 7(1):114, 2006.

[73] David T Jones. Protein secondary structure prediction based on position-specific scoring matrices. *Journal of molecular biology*, 292(2):195–202, 1999.

[74] Frank Jühling, Mario Mörl, Roland K Hartmann, Mathias Sprinzl, Peter F Stadler, and Joern Pütz. trnadb 2009: compilation of trna sequences and trna genes. *Nucleic acids research*, 37(suppl_1):D159–D162, 2008.

[75] David R Kelley, Jasper Snoek, and John L Rinn. Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks. *Genome research*, 26(7):990–999, 2016.

[76] Scott E Kern, Kenneth W Kinzler, Arthur Bruskin, David Jarosz, Paula Friedman, Carol Prives, and Bert Vogelstein. Identification of p53 as a sequence-specific dna-binding protein. *Science*, 252(5013):1708–1711, 1991.

[77] Eeva Kettunen, Sisko Anttila, Jouni K Seppänen, Antti Karjalainen, Henrik Edgren, Irmeli Lindström, Reijo Salovaara, Anna-Maria Nissén, Jarmo Salo, Karin Mattson, et al. Differentially expressed genes in nonsmall cell lung cancer: expression profiling of cancer-related genes in squamous cell lung cancer. *Cancer genetics and cytogenetics*, 149(2):98–106, 2004.

[78] Ghazaleh Khodabandelou, Etienne Routhier, and Julien Mozziconacci. Genome functional annotation using deep convolutional neural networks. *bioRxiv*, page 330308, 2018.

[79] B Kim and F Doshi-Velez. Interpretable machine learning: the fuss, the concrete and the questions. *ICML Tutorial on interpretable machine learning*, 2017.

[80] Kouichi Kimura, Ai Wakamatsu, Yutaka Suzuki, Toshio Ota, Tetsuo Nishikawa, Riu Yamashita, Jun-ichi Yamamoto, Mitsuo Sekine, Katsuki Tsuritani, Hiroyuki Wakaguri, et al. Diversification of transcriptional modulation: large-scale identification and characterization of putative alternative promoters of human genes. *Genome research*, 16(1):55–65, 2006.

[81] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada, 1995.

[82] Peter K Koo and Sean R Eddy. Representation learning of genomic sequence motifs with convolutional neural networks. *BioRxiv*, page 362756, 2018.

[83] Konstantina Kourou, Themis P Exarchos, Konstantinos P Exarchos, Michalis V Karamouzis, and Dimitrios I Fotiadis. Machine learning applications in cancer prognosis and prediction. *Computational and structural biotechnology journal*, 13:8–17, 2015.

[84] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[85] Rachael C Kuintzle, Eileen S Chow, Tara N Westby, Barbara O Gvakharia, Jadwiga M Giebultowicz, and David A Hendrix. Circadian deep sequencing reveals stress-response genes that adopt robust rhythmic expression during aging. *Nature communications*, 8:14529, 2017.

[86] Miron B Kursa, Witold R Rudnicki, et al. Feature selection with the boruta package, 2010.

[87] Jack Lanchantin, Ritambhara Singh, Beilun Wang, and Yanjun Qi. Deep motif dashboard: Visualizing and understanding genomic sequences using deep neural networks. In *PACIFIC SYMPOSIUM ON BIOCOMPUTING 2017*, pages 254–265. World Scientific, 2017.

[88] Niels Larsen, Tore Samuelsson, and Christian Zwieb. The signal recognition particle database (srpdb). *Nucleic acids research*, 26(1):177–178, 1998.

[89] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 609–616. ACM, 2009.

[90] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *CoRR*, abs/1609.06570, 2016.

[91] Neocles B Leontis and Eric Westhof. Geometric nomenclature and classification of rna base pairs. *Rna*, 7(4):499–512, 2001.

[92] Hao Li, Beiqin Yu, Jianfang Li, Liping Su, Min Yan, Jun Zhang, Chen Li, Zhenggang Zhu, and Bingya Liu. Characterization of differentially expressed genes involved in pathways associated with gastric cancer. *PloS one*, 10(4):e0125013, 2015.

[93] Jinyan Li, Huiqing Liu, See-Kiong Ng, and Limsoon Wong. Discovery of significant rules for classifying cancer diagnosis data. *Bioinformatics*, 19(suppl 2):ii93–ii102, 2003.

[94] Weizhong Li and Adam Godzik. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22(13):1658–1659, 2006.

[95] Weizhong Li and Adam Godzik. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22:1658–1659, 2006.

[96] Biao Liu, Jessica L Childs-Disney, Brent M Znosko, Dan Wang, Mohammad Fallahi, Steven M Gallo, and Matthew D Disney. Analysis of secondary structural elements in human microrna hairpin precursors. *BMC bioinformatics*, 17(1):112, 2016.

[97] Marina Lizio, Jayson Harshbarger, Hisashi Shimoji, Jessica Severin, Takeya Kasukawa, Serkan Sahin, Imad Abugessaisa, Shiro Fukuda, Fumi Hori, Sachi Ishikawa-Kato, et al. Gateways to the fantom5 promoter level mammalian expression atlas. *Genome biology*, 16(1):22, 2015.

[98] Xiang-Jun Lu, Harmen J Bussemaker, and Wilma K Olson. Dssr: an integrated software tool for dissecting the spatial structure of rna. *Nucleic acids research*, 43(21):e142–e142, 2015.

[99] Shai Lubliner, Leeat Keren, and Eran Segal. Sequence features of yeast and human core promoters that are predictive of maximal promoter activity. *Nucleic acids research*, 41(11):5569–5581, 2013.

[100] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4765–4774, 2017.

[101] Rune B Lyngsø and Christian NS Pedersen. Rna pseudoknot prediction in energy-based models. *Journal of computational biology*, 7:409–427, 2000.

[102] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[103] Mark Maienschein-Cline, Jie Zhou, Kevin P White, Roger Sciammas, and Aaron R Dinner. Discovering transcription factor regulatory targets using gene expression and binding data. *Bioinformatics*, 28(2):206–213, 2012.

[104] David H Mathews, Matthew D Disney, Jessica L Childs, Susan J Schroeder, Michael Zuker, and Douglas H Turner. Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of rna secondary structure. *Proceedings of the National Academy of Sciences*, 101(19):7287–7292, 2004.

[105] David H Mathews, Jeffrey Sabina, Michael Zuker, and Douglas H Turner. Expanded sequence dependence of thermodynamic parameters improves prediction of rna secondary structure. *Journal of molecular biology*, 288(5):911–940, 1999.

[106] David H Mathews, Jeffrey Sabina, Michael Zuker, and Douglas H Turner. Expanded sequence dependence of thermodynamic parameters improves prediction of rna secondary structure. *Journal of molecular biology*, 288:911–940, 1999.

[107] Greg Matlashewski, Peter Lamb, David Pim, Jim Peacock, Lionel Crawford, and S Benchimol. Isolation and characterization of a human p53 cdna clone: expression of the human p53 gene. *The EMBO journal*, 3(13):3257, 1984.

[108] Seyyid Ahmed Medjahed, Tamazouzt Ait Saadi, and Abdelkader Benyettou. Breast cancer diagnosis by using k-nearest neighbor with different distances and classification rules. *International Journal of Computer Applications*, 62(1), 2013.

[109] Dirk Metzler and Markus E Nebel. Predicting rna secondary structures with pseudoknots by mcmc sampling. *Journal of mathematical biology*, 56:161–181, 2008.

[110] Francois Michel and Eric Westhof. Modelling of the three-dimensional architecture of group i catalytic introns based on comparative sequence analysis. *Journal of molecular biology*, 216(3):585–610, 1990.

[111] Paul JA Michiels, Alexandra AM Versleijen, Paul W Verlaan, Cornelis WA Pleij, Cornelis W Hilbers, and Hans A Heus. Solution structure of the pseudoknot of srv-1 rna, involved in ribosomal frameshifting. *Journal of molecular biology*, 310:1109–1123, 2001.

[112] Venkatesh L Murthy and George D Rose. Rnabase: an annotated database of rna structures. *Nucleic Acids Research*, 31(1):502–504, 2003.

[113] Jennifer S Myers, Ariana K von Lersner, Charles J Robbins, and Qing-Xiang Amy Sang. Differentially expressed genes and signature pathways of human prostate cancer. *PloS one*, 10(12):e0145322, 2015.

[114] Eric P Nawrocki, Sarah W Burge, Alex Bateman, Jennifer Daub, Ruth Y Eberhardt, Sean R Eddy, Evan W Floden, Paul P Gardner, Thomas A Jones, John Tate, et al. Rfam 12.0: updates to the rna families database. *Nucleic acids research*, 43(D1):D130–D137, 2014.

[115] Cancer Genome Atlas Research Network et al. Comprehensive molecular characterization of clear cell renal cell carcinoma. *Nature*, 499(7456):43–49, 2013.

[116] Paul L Nixon, Anupama Rangan, Y-G Kim, Alexander Rich, David W Hoffman, Mirko Hennig, and David P Giedroc. Solution structure of a luteoviral p1–p2 frameshifting mrna pseudoknot. *Journal of molecular biology*, 322:621–633, 2002.

[117] Shuhei Noguchi, Takahiro Arakawa, Shiro Fukuda, Masaaki Furuno, Akira Hasegawa, Fumi Hori, Sachi Ishikawa-Kato, Kaoru Kaida, Ai Kaiho, Mutsumi Kanamori-Katayama, et al. Fantom5 cage profiles of human and mouse samples. *Scientific data*, 4:170112, 2017.

[118] Ruth Nussinov and Ann B Jacobson. Fast algorithm for predicting the secondary structure of single-stranded rna. *Proceedings of the National Academy of Sciences*, 77(11):6309–6313, 1980.

[119] Uwe Ohler. Identification of core promoter modules in drosophila and their application in accurate transcription start site prediction. *Nucleic acids research*, 34(20):5943–5950, 2006.

[120] Trevor J Parry, Joshua WM Theisen, Jer-Yuan Hsu, Yuan-Liang Wang, David L Corcoran, Moriah Eustice, Uwe Ohler, and James T Kadonaga. The tct motif, a key component of an rna polymerase ii transcription system for the translational machinery. *Genes & development*, 24(18):2013–2018, 2010.

[121] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[122] Rouaïda Cavin Périer, Viviane Praz, Thomas Junier, Claude Bonnard, and Philipp Bucher. The eukaryotic promoter database (epd). *Nucleic acids research*, 28(1):302–303, 2000.

[123] Adam E Peritz, Ryszard Kierzek, Naoki Sugimoto, and Douglas H Turner. Thermodynamic study of internal loops in oligoribonucleotides: symmetric loops are more stable than asymmetric loops. *Biochemistry*, 30(26):6428–6436, 1991.

[124] DM Powers. Proceedings of the joint conferences on new methods in language processing and computational natural language learning. Association for Computational Linguistics Stroudsburg, PA, 1998.

[125] Max Pumperla. Hyperas: A very simple wrapper for convenient hyperparameter optimization. available: https://github.com/maxpumperla/hyperas. 2017.

[126] Tomasz Puton, Lukasz P Kozlowski, Kristian M Rother, and Janusz M Bujnicki. Comparna: a server for continuous benchmarking of automated methods for rna secondary structure prediction. *Nucleic acids research*, 41(7):4307–4323, 2013.

[127] Tomasz Puton, Lukasz P Kozlowski, Kristian M Rother, and Janusz M Bujnicki. Comparna: a server for continuous benchmarking of automated methods for rna secondary structure prediction. *Nucleic acids research*, 41:4307–4323, 2013.

[128] Ying Qian, Yu Zhang, Bingyu Guo, Shasha Ye, Yuzhu Wu, and Jiongmin Zhang. An improved promoter recognition model using convolutional neural network. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, volume 1, pages 471–476. IEEE, 2018.

[129] Daniel Quang and Xiaohui Xie. Danq: a hybrid convolutional and recurrent deep neural network for quantifying the function of dna sequences. *Nucleic acids research*, 44(11):e107–e107, 2016.

[130] Baharak Rastegari and Anne Condon. Linear time algorithm for parsing rna secondary structure. In *International Workshop on Algorithms in Bioinformatics*, pages 341–352. Springer, 2005.

[131] Toolika Rastogi, Tara L Beattie, Joan E Olive, and Richard A Collins. A long-range pseudoknot is required for activity of the neurospora vs ribozyme. *The EMBO journal*, 15:2820, 1996.

[132] SVG Reddy, K Thammi Reddy, V Valli Kumari, and Kamadi VSRP Varma. An svm based approach to breast cancer classification using rbf and polynomial kernel functions with varying arguments. *International Journal of Computer Science and Information Technologies*, 5(4):5901–5904, 2014.

[133] George P Rédei. Tetraloop. *Encyclopedia of Genetics, Genomics, Proteomics and Informatics*, pages 1953–1954, 2008.

[134] Jens Reeder, Peter Steffen, and Robert Giegerich. pknotsrg: Rna pseudoknot folding including near-optimal structures and sliding windows. *Nucleic acids research*, 35:W320–W324, 2007.

[135] Payam Refaeilzadeh, Lei Tang, and Huan Liu. Cross-validation. pages 532–538, 2009.

[136] Elena Rivas and Sean R Eddy. A dynamic programming algorithm for rna structure prediction including pseudoknots. *Journal of molecular biology*, 285:2053–2068, 1999.

[137] Peter W Rose, Andreas Prlić, Ali Altunkaya, Chunxiao Bi, Anthony R Bradley, Cole H Christie, Luigi Di Costanzo, Jose M Duarte, Shuchismita Dutta, Zukang Feng, et al. The rcsb protein data bank: integrative view of protein, gene and 3d structural information. *Nucleic acids research*, page gkw1000, 2016.

[138] David Saad. Online algorithms and stochastic approximations. *Online Learning.*

[139] Albin Sandelin, Wynand Alkema, Pär Engström, Wyeth W Wasserman, and Boris Lenhard. Jaspar: an open-access database for eukaryotic transcription factor binding profiles. *Nucleic acids research*, 32(suppl_1):D91–D94, 2004.

[140] Kengo Sato, Yuki Kato, Michiaki Hamada, Tatsuya Akutsu, and Kiyoshi Asai. Ipknot: fast and accurate prediction of rna secondary structures with pseudoknots using integer programming. *Bioinformatics*, 27:i85–i93, 2011.

[141] Eric E Schadt, John Lamb, Xia Yang, Jun Zhu, Steve Edwards, Debraj GuhaThakurta, Solveig K Sieberts, Stephanie Monks, Marc Reitman, Chunsheng Zhang, et al. An integrative genomics approach to infer causal associations between gene expression and disease. *Nature genetics*, 37(7):710–717, 2005.

[142] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *International Conference on Artificial Neural Networks*, pages 583–588. Springer, 1997.

[143] Jacob Schreiber, Ritambhara Singh, Jeffrey Bilmes, and William Stafford Noble. A pitfall for machine learning methods aiming to predict across cell types. *bioRxiv*, page 512434, 2019.

[144] Martin J Serra, Theresa J Axenson, and Douglas H Turner. A model for the stabilities of rna hairpins based on a study of the sequence dependence of stability for hairpins of six nucleotides. *Biochemistry*, 33(47):14289–14296, 1994.

[145] Martin J Serra, Matthew H Lyttle, Theresa J Axenson, Calvin A Schadt, and Douglas H Turner. Rna hairpin loop stability depends on closing base pair. *Nucleic acids research*, 21(16):3845–3849, 1993.

[146] KM Shabana, KA Abdul Nazeer, Meeta Pradhan, and Mathew Palakal. A computational method for drug repositioning using publicly available gene expression data. *BMC bioinformatics*, 16(17):1, 2015.

[147] Ling X Shen and Ignacio Tinoco Jr. The structure of an rna pseudoknot that causes efficient frameshifting in mouse mammary tumor virus. *Journal of molecular biology*, 247:963–978, 1995.

[148] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[149] Sandra Smit, Kristian Rother, Jaap Heringa, and Rob Knight. From knotted to nested rna structures: a variety of computational methods for pseudoknot removal. *RNA*, 14(3):410–416, 2008.

[150] Sandra Smit, Michael Yarus, and Rob Knight. Natural selection is not required to explain universal compositional patterns in rrna secondary structure categories. *RNA*, 12(1):1–14, 2006.

[151] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[152] Victor Solovyev and Ramzan Umarov. Prediction of prokaryotic and eukaryotic promoters using convolutional deep learning neural networks. *arXiv preprint arXiv:1610.00121*, 2016.

[153] Jana Sperschneider and Amitava Datta. Dotknot: pseudoknot prediction using the probability dot plot under a refined energy model. *Nucleic acids research*, 38:e103–e103, 2010.

[154] Jana Sperschneider, Amitava Datta, and Michael J Wise. Heuristic rna pseudoknot prediction including intramolecular kissing hairpins. *Rna*, 17(1):27–38, 2011.

[155] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[156] David W Staple and Samuel E Butcher. Pseudoknots: Rna structures with diverse functions. *PLoS Biol*, 3:e213, 2005.

[157] Gary D Stormo. Dna binding sites: representation and discovery. *Bioinformatics*, 16(1):16–23, 2000.

[158] P Svoboda and A Di Cara. Hairpin rna: a secondary structure of primary importance. *Cellular and Molecular Life Sciences CMLS*, 63(7-8):901–908, 2006.

[159] Hazuki Takahashi, Sachi Kato, Mitsuyoshi Murata, and Piero Carninci. Cage (cap analysis of gene expression): a protocol for the detection of promoter and transcriptional networks. In *Gene regulatory networks*, pages 181–200. Springer, 2012.

[160] Makio Tamura, Donna K Hendrix, Peter S Klosterman, Nancy RB Schimmelman, Steven E Brenner, and Stephen R Holbrook. Scor: Structural classification of rna, version 2.0. *Nucleic acids research*, 32(suppl_1):D182–D184, 2004.

[161] Aik Choon Tan and David Gilbert. Ensemble machine learning on gene expression data for cancer classification. 2003.

[162] Cheng C et al. Tan J, Ung M. Unsupervised feature construction and knowledge extraction from genome-wide assays of breast cancer with denoising autoencoders, 2015.

[163] Carla A Theimer, Craig A Blois, and Juli Feigon. Structure of the human telomerase rna pseudoknot reveals conserved tertiary interactions essential for function. *Molecular cell*, 17:671–682, 2005.

[164] Craig Tuerk, Peter Gauss, Claude Thermes, Duncan R Groebe, Margit Gayle, Nancy Guild, Gary Stormo, Yves d'Aubenton Carafa, Olke C Uhlenbeck, and Ignacio Tinoco. Cuucgg hairpins: extraordinarily stable rna secondary structures associated with various biochemical processes. *Proceedings of the National Academy of Sciences*, 85(5):1364–1368, 1988.

[165] Rahul Tyagi and David H Mathews. Predicting helical coaxial stacking in rna multibranch loops. *Rna*, 13(7):939–951, 2007.

[166] Yasuo Uemura, Aki Hasegawa, Satoshi Kobayashi, and Takashi Yokomori. Tree adjoining grammars for rna structure prediction. *Theoretical computer science*, 210:277–303, 1999.

[167] Ramzan Umarov, Hiroyuki Kuwahara, Yu Li, Xin Gao, and Victor Solovyev. Promid: human promoter prediction by deep learning. *arXiv preprint arXiv:1810.01414*, 2018.

[168] Ramzan Umarov, Hiroyuki Kuwahara, Yu Li, Xin Gao, and Victor Solovyev. Promoter analysis and prediction in the human genome using sequence-based deep learning models. *Bioinformatics*, 2019.

[169] FHD Van Batenburg, Alexander P Gultyaev, and Cornelis WA Pleij. Pseudobase: structural information on rna pseudoknots. *Nucleic acids research*, 29:194–195, 2001.

[170] Pamela L Vanegas, Graham A Hudson, Amber R Davis, Shannon C Kelly, Charles C Kirkpatrick, and Brent M Znosko. Rna cossmos: characterization of secondary structure motifs—a searchable database of secondary structure motifs in rna three-dimensional structures. *Nucleic acids research*, 40(D1):D439–D444, 2011.

[171] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[172] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[173] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 11:3371–3408, 2010.

[174] Sun-Chong Wang. Artificial neural network. In *Interdisciplinary Computing in Java Programming*, pages 81–100. Springer, 2003.

[175] Stefan Washietl, Sebastian Will, David A Hendrix, Loyal A Goff, John L Rinn, Bonnie Berger, and Manolis Kellis. Computational analysis of noncoding rnas. *Wiley Interdisciplinary Reviews: RNA*, 3:759–778, 2012.

[176] John N Weinstein, Eric A Collisson, Gordon B Mills, Kenna R Mills Shaw, Brad A Ozenberger, Kyle Ellrott, Ilya Shmulevich, Chris Sander, Joshua M Stuart, Cancer Genome Atlas Research Network, et al. The cancer genome atlas pan-cancer analysis project. *Nature genetics*, 45(10):1113–1120, 2013.

[177] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[178] John C Wootton and Scott Federhen. [33] analysis of compositionally biased regions in sequence databases. In *Methods in enzymology*, volume 266, pages 554–571. Elsevier, 1996.

[179] Johnny C Wu, David P Gardner, Stuart Ozer, Robin R Gutell, and Pengyu Ren. Correlation of rna secondary structure statistics with thermodynamic stability and applications to folding. *Journal of molecular biology*, 391(4):769–783, 2009.

[180] A Xayaphoummine, T Bucher, F Thalmann, and H Isambert. Prediction and statistics of pseudoknots in rna structures using exactly clustered stochastic simulations. *Proceedings of the National Academy of Sciences*, 100(26):15310–15315, 2003.

[181] Jiangchun Xu, John A Stolk, Xinqun Zhang, Sandra J Silva, Raymond L Houghton, Masazumi Matsumura, Thomas S Vedvick, Kevin B Leslie, Roberto Badaro, and Steven G Reed. Identification of differentially expressed genes in human prostate cancer using subtraction and microarray. *Cancer research*, 60(6):1677–1682, 2000.

[182] Riu Yamashita, Nuankanya P Sathira, Akinori Kanai, Kousuke Tanimoto, Takako Arauchi, Yoshiaki Tanaka, Shin-ichi Hashimoto, Sumio Sugano, Kenta Nakai, and Yutaka Suzuki. Genome-wide characterization of transcriptional start sites in humans by integrative transcriptome analysis. *Genome research*, 21(5):775–789, 2011.

[183] Chuhu Yang, Eugene Bolotin, Tao Jiang, Frances M Sladek, and Ernest Martinez. Prevalence of the initiator over the tata box in human and yeast genes and identification of dna motifs enriched in human tata-less core promoters. *Gene*, 389(1):52–65, 2007.

[184] Huanwang Yang, Fabrice Jossinet, Neocles Leontis, Li Chen, John Westbrook, Helen Berman, and Eric Westhof. Tools for the automatic identification and classification of rna base pairs. *Nucleic acids research*, 31(13):3450–3460, 2003.

[185] Ka Yee Yeung and Walter L. Ruzzo. Principal component analysis for clustering gene expression data. *Bioinformatics*, 17(9):763–774, 2001.

[186] Shay Zakov, Yoav Goldberg, Michael Elhadad, and Michal Ziv-Ukelson. Rich parameterization improves rna structure prediction. *Journal of Computational Biology*, 18:1525–1542, 2011.

[187] Teng Zhou, Yipeng Du, and Taotao Wei. Transcriptomic analysis of human breast cancer cells reveals differentially expressed genes and related cellular functions and pathways in response to gold nanorods. *Biophysics Reports*, 1(2):106–114, 2015.

[188] Xin Zhou, Brett Maricque, Mingchao Xie, Daofeng Li, Vasavi Sundaram, Eric A Martin, Brian C Koebbe, Cydney Nielsen, Martin Hirst, Peggy Farnham, et al. The human epigenome browser at washington university. *Nature methods*, 8(12):989, 2011.

[189] George Kingsley Zipf. *The psycho-biology of language: An introduction to dynamic philology.* Routledge, 2013.

[190] Christian Zwieb, Jan Gorodkin, Bjarne Knudsen, Jody Burks, and Jacek Wower. tmrdb (tmrna database). *Nucleic acids research*, 31(1):446–447, 2003.

APPENDICES

Appendix A: Interpretability through Exploratory Data Analysis and Large-scale Automated Annotation of RNA Secondary Structures

A.1 Definition of a Segment

Let's consider an RNA sequence r such that $r[i] \in \{A, C, G, U\}$. The length of r given by $|r|$. We can define a structure S of r to be a set of pairs of positions of r called base-pairs. That is, $(i, j) \in S$, where $1 \leq i < j \leq |r|$, and the nucleotides $r[i]$ and $r[j]$ form a base pair, and that they are typically required to be such that $j - i > d_{min}$ for some d_{min} .

It can be helpful to define a function $\phi(i)$ that maps a position i to the position j to which it is paired. Therefore, $\phi(i) = j$ and $\phi(j) = i$ for each $(i, j) \in S$. For unpaired nucleotides i , $\phi(i) = 0$.

We can define the "next paired nucleotide" $n(i)$ is the next nucleotide in the 3' direction that forms a base pair:

$$n(i) = \underset{k : k > i, \phi(k) \neq 0}{argmin} k - i$$

and "previous paired nucleotide" as, and $p(i)$ is the previous paired nucleotide in the 5' direction.

$$p(i) = \underset{k : k < i, \phi(k) \neq 0}{argmin} i - k$$

The functions $n(i)$ and $p(i)$ are inverse functions, meaning $n(p(i)) = i$. We further add to these functions that they return 0 when there is no next paired base or previous base.

Two base pairs $(i, j) \in S$ and $(i', j') \in S$ are said to be its "adjacent base pairs" if either

$n(i) = i'$ and $p(j) = j'$

$p(i) = i'$ and $n(j) = j'$

A segment $\sigma \subseteq S$ is a set of adjacent base pairs $\beta_1, \beta_2, \ldots, \beta_{|\sigma|}$ such that β_i is adjacent to β_{i+1} for $i = 1, \ldots, |\sigma| - 1$. A base pair cannot occupy more than one segment. A segment is different from a stem because it can contain bulges and internal loops.

We can define an ordering on the segments based on the numerical ordering of the lowest position of each segment. In this way, segments can be numbered from 5' to 3', such that $\sigma_1 < \sigma_2 < \ldots < \sigma_N$.

Here is the algorithm for identification of segments:

Algorithm 1 IDENTIFYSEGMENTS(ϕ), is the algorithm for identification of segments.

```
1:  procedure IdentifySegments($\phi$)
        # The input the base pair function $\phi(i)$ for an RNA
2:      $\mathcal{S} \leftarrow \{\}$                                   ▷ the set of segments
3:      $(\mathcal{F}, \mathcal{L}) \leftarrow getFirstAndLast(\phi)$   ▷ get the first and last paired position
4:      $i \leftarrow \mathcal{F}$
5:      $j \leftarrow \phi(i)$
6:      while $\mathcal{F} \leq i \leq \mathcal{L}$ do
7:          $\sigma \leftarrow \{\}$                                    ▷ initialize the segment to the empty set
8:          if $i < j$ then
9:              $\sigma \leftarrow \sigma \cup \{(i, j)\}$              ▷ add base pair to segment
10:             $INSEGMENT \leftarrow True$
11:         while $INSEGMENT$ do
12:             if $n(i) > 0 and p(j) > 0$ then
13:                 if $\phi(n(i)) = p(i) and n(i) < p(j)$ then
14:                     $\sigma \leftarrow \sigma \cup \{(n(i), p(j)\}$  ▷ add the next base pair to segment
15:                     $i \leftarrow n(i)$                         ▷ update the $i$ position to the next base pair
16:                     $j \leftarrow p(j)$                         ▷ update the $j$ position to next pair
17:                 else
18:                     $INSEGMENT \leftarrow False$
19:                     $\mathcal{S} \leftarrow \mathcal{S} \cup \{\sigma\}$   ▷ store the segment
20:             else
21:                 $INSEGMENT \leftarrow False$
22:                 $\mathcal{S} \leftarrow \mathcal{S} \cup \{\sigma\}$       ▷ store the segment
23:         $i \leftarrow n(i)$
24:         $j \leftarrow \phi(i)$
```

Algorithm 2 MWISPathGraph(c), a dynamic programming algorithm to search for Maximum Weighted Independent Subset on path graphs

```
1: procedure MWISPathGraph($c, v(.), w(.)$)
        # input is a connected component where all the vertices are in a path (Figure 2.2)
        #$w()$ is an array of the weights (number of base pairs) $w(i)$ of vertex $i$
        #$v()$ is an array of the vertex labels $v(i)$ of vertex $i$
        #Set S stores the maximum-weighted independent set
        #vertices assumed to be numbered $1, ..., |c|$
2:     $\mathcal{S} \leftarrow \{\}$
3:     $Wmax = Array$                ▷ Weight array $Wmax$ stores maximum weight so far
4:     $Wmax[0] \leftarrow 0$
5:     $Wmax[1] \leftarrow w(1)$
        # First, assign terms to weight array
6:     for $i \in \{2, ..., |c|\}$ do
7:         $Wmax[i] = max(Wmax[i-1], Wmax[i-2] + w(i))$
        # Now, traverse weight array, extract maximum subset
8:     $i \leftarrow |c|$
9:     while $i \geq 1$ do
10:        if $i = 2$ then
11:            if $w(i) = w(i-1)$ then                ▷ special case for first two vertices
12:                $\mathcal{S} \leftarrow \mathcal{S} \cup \{max(v(i), v(i-1))\}$    ▷ when equally weighted, store more $3'$
    vertex
13:                $i \leftarrow i - 1$
14:        if $i = |c|$ then
15:            if $w(i) = w(i-1)$ then                ▷ special case for last two vertices
16:                if $Wmax[i] = Wmax[i-2]$ then
17:                    if $v(i) > v(i-1)$ then
18:                        $\mathcal{S} \leftarrow \mathcal{S} \cup \{v(i)\}$
19:                        $i \leftarrow i - 2$
20:        if $Wmax[i] = Wmax[i-1]$ then
21:            $i \leftarrow i - 1$
22:        else
23:            $\mathcal{S} \leftarrow \mathcal{S} \cup \{v(i)\}$
24:            $i \leftarrow i - 2$
```

Algorithm 3 PK-DETECTION algorithm identifies segments that are PKs. These segments contain the smallest set of base pairs that when removed result in a PK-free structure. This algorithm is a heuristic approach for identification of the maximum-weighted independent set of the PK graph. For path graphs, MWISPathGraph() is called, which is an exact solution for path graphs.

```
1:  procedure PK_Detection($\mathcal{G}$)
2:      while $PK(\mathcal{G})$ do
3:          $CC \leftarrow ConnectedComponents(\mathcal{G})$    ▷ Collect all connected components in the PK-Graph
4:          for $c \in CC$ do
5:              if $|c| = 2$ then
6:                  $(v1, v2) \leftarrow c$    ▷ If only two vertices, vertex with smallest weight is PK
7:                  if $w(v1) < w(v2)$ then
8:                      $Delete(\mathcal{G}, v1)$
9:                  else if $w(v1) > w(v2)$ then
10:                     $Delete(\mathcal{G}, v2)$
11:                 else    ▷ vertices have equal weight, $5'$ vertex is PK (heuristic)
12:                     if $v1 < v2$ then
13:                         $Delete(\mathcal{G}, v1)$
14:                     else
15:                         $Delete(\mathcal{G}, v2)$
16:             else if $pathGraph(c)$ then
17:                 if $|c| = 3$ and $w(v2) = w(v1) + w(v3)$ then
                                                ▷ check special case of 3-vertex path graph
18:                     $(v1, v2, v3) \leftarrow c$    ▷ tie: choose middle vertex as PK to make kissing hairpin
19:                     $Delete(\mathcal{G}, v2)$
20:                 else
21:                     $\mathcal{S} \leftarrow$MWISPathGraph($c$)
                                    ▷ delete the complement of max weighted independent set
22:                     $Delete(\mathcal{G}, \{1, ..., |c|\} \setminus \mathcal{S})$
23:             else    ▷ complex non-path graph with $> 2$ nodes
24:                 $minW \leftarrow minWeight(c)$    ▷ vertices with lowest weight
25:                 $maxD \leftarrow maxDegree(c)$    ▷ vertices with highest degree
26:                 for $v \in maxD$ do
27:                     $score(v) = neighborWeightSum(v) – w(v)$
28:                 $v^* = maxScore(c)$    ▷ identify vertex with maximum score
29:                 if $score(v^*) > 0$ then    ▷ highest degree vertex is PK
30:                     $Delete(\mathcal{G}, v^*)$    ▷ Its neighbors collectively have more weight (base pairs)
31:                 else    ▷ no max-degree vertex has positive score
32:                     $\{v\} \leftarrow minW$    ▷ minimum-weight vertex is PK
33:                     $Delete(\mathcal{G}, v)$
```

Algorithm 4 MaxFirstPagePairs($\mathcal{S}, \mathcal{L}$) algorithm uses a bottom-up CKY-approach to identify a minimum set of base pairs to produce a PK-free structure. The output is a dot-bracket sequence with PK base pairs removed.

```
1: procedure MaxFirstPagePairs($\mathcal{S}, \mathcal{L}$)
       #input is a set of base pairs S for an RNA of length $\mathcal{L}$.
2:     $score = 2D$ Array      ▷ matrix score stores the number of pairs on each span $[i, j]$
3:     $backptr = 2D$ Array                    ▷ backptr for the corresponding $[i, j]$ span
4:     for $i \in \{0, ..., \mathcal{L}\}$ do                  ▷ initialize the $length = 0, 1$ spans
5:         $score[i][i] \leftarrow 0$
6:         $score[i][i+1] \leftarrow 0$
                                                                  ▷ bottom-up CKY
7:     for $span \in \{2, ..., \mathcal{L}\}$ do
8:         for $i \in \{0, ..., \mathcal{L} - span\}$ do
9:             $j \leftarrow i + span$
10:            if $(i, j-1) \in \mathcal{S}$ then                           ▷ pair$(i, j-1)$
11:                $score[i][j] \leftarrow 1 + score[i+1][j-1]$
12:                $backptr[i][j] \leftarrow PAIR, -1)$
13:            else
14:                $score[i][j] \leftarrow 0$
                                                    ▷ loop over all split points
15:            for $k \in \{i+1, ..., j-1\}$ do
16:                if $score[i][k] + score[k][j] > score[i][j]$ then
17:                    $score[i][j] \leftarrow score[i][k] + score[k][j]$
18:                    $backptr[i][j] \leftarrow (SPLIT, k)$
19:    return BACKTRACK$(i, j)$
```

Algorithm 5 BACKTRACK(i, j) algorithm uses dynamic programming to recursively produce a dot-bracket sequence. Here, "+" refers to string-concatenation, and "."$\times n$ copies the "." character n times.

```
1: procedure backtrack$(i, j)$              ▷ recursively backtrack to get structures
2:     if $score[i][j] = 0$ then
3:         return "."$\times(j - i)$
4:     else if $backptr[i][j][0] = PAIR$ then
5:         return "(" + BACKTRACK$(i+1, j-1)$+ ")"
6:     else if $backptr[i][j][0] = SPLIT$ then
7:         $k = backptr[i][j][1]$
8:         return BACKTRACK$(i, k)$+ BACKTRACK$(k, j)$
```

A.2 Pseudoknots

A pseudoknot (PK) is characterized by crossing base pairs called a PK-quartet. A PK-quartet consists of two base pairs (i, j) and (i', j') in S that satisfies the PK-ordering such that either $i < i' < j < j'$ or $i' < i < j' < j$.

Theorem: If any two segments σ and τ that each have a base pair that comprise a PK-quartet, then every base pair in σ forms a PK-quartet with every base pair in τ . This follows from the definition of a segment. Consider $(i, j) \in \sigma$ and $(i', j') \in \tau$ such that they form a PK-quartet, and without loss of generality, assume that $i < i' < j < j'$.

Proof: Consider $(k, l) \in \sigma$ to be a adjacent to (i, j) . Let's assume that $n(i) = k$ and $p(j) = l$, but a similar proof follows in the other direction. It must be the case that either $k < i'$ or $i' < k$. Because $i < i'$, the first case, $i^{'} < k$, requires that the segment σ has adjacent nucleotides i and k with an intervening paired nucleotide i' that is not in σ , which would violate the definition of adjacent nucleotides. Therefore, $k < i'$. Similarly, it can be shown that $i' < l$. Finally, since $n(l) = j$, and $j < j'$, it follows that that $l < j'$. Therefore, $k < i' < l < j'$. This approach can be continued for all base pairs in σ and τ .

For each structure, we define a subset of segments as the PK-segments, which are the minimum set of segments that can be removed to produce a PK-free structure. We annotate different PK-segments as a set of base pairs connecting what would otherwise be loops if the PK-segments were removed. This can be achieved through the creation of PK-free segment graph, followed by annotation of each loop in this substructure. In general, the base pairs of a PK-segment can form PK-quartets with base pairs from many different segments. We can process these by forming a PK-segment graph.

A PK-segment graph is an undirected graph with the nodes corresponding to segments, and the edges connecting segments that have base pairs that form PK-quartets with each other. As proven above, this is equivalent to checking if one pair of base pairs has this property. Clearly, the vertices of the PK-segment graph are a subset of the vertices in the segment graph of a structure because only segments that form PKs with other segments can be included. The PK-graph will in general consist of several connected components. It is not sufficient to remove, for each connected component of this graph, the segment with the smallest weight (smallest number of base pairs), and labeled a PK-segment. This fails on many examples. We used the following algorithm to satisfactorily process all structures in bpRNA-1m, using the following functions:

$ConnectedComponents(G)$ = list of connected components c in the graph G .

$w(v)$ = the weight of the vertex v in number of base pairs.
$Delete(G,v)$ = delete the vertex v from graph G (v is labeld as a PK).
$pathGraph(c)$ = returns True if the connected component c is a path graph, False otherwise.
$neighborWeightSum(v)$ = sum of the weights $w(u)$ for each neighbor u of vertex v .
$score(v) = neighborWeightSum(v) - w(v)$
$MWISPathGraph(c)$ = Dynamic programming algorithm for Maximum Weighted Independent Subset on path graphs, defined here:

A.3 bpRNA-1m Database Schema

The full database schema for bpRNA-1m is presented in Figure A.1. There are 3 categories of tables in the database. (1) Main table: the RNA table (blue table in the diagram) is the main table in our database, which means all other tables are connected to this table even directly or indirectly and it is the core of our database schema. This table has all the information about an RNA such as bpRNA name, length of the RNA sequence, the original source and identification number, organism, lineage, and etc. (2) General tables: such tables have one to one relationship with the main table (green tables). For example, there is only one RNA type is associated with each RNA in the main table. Reference database, RNA type, method, and validation are the tables in this category. (3) Structural tables: These types encompass all the structural details about any types of loop (hairpins, bulges, internal, and multi), unpaired regions, stems, segments, and pseudoknots (red and yellow colored tables). The information that these tables provide are base pairs, positions, length of the structure, the loop sequence, whether a loop has pseudoknot, and etc.

In case of Internal loops, multiloops, and pseudoknots there was a need for additional tables that are indirectly connected to the main tables and contain data from the structures that have more than one loop. For example, there are several loops are involved in multiloops structures to save the information of every branch we need an additional table (MultiloopBranch table), which is connected to Multioop table with a unique id of that specific structure. These tables are colored yellow in the database diagram.

There is a unique back-end numerical RNA-ID for each RNA in bpRNA-1m, which is used for accessing data from all other tables. There is also a unique RNA-Name that is used for accessing the data on the webserver through our interactive PHP website. We have three different types of tables in the database. The first is the main table, which is connected to all other tables directly or indirectly. The second type are general tables with data that can be shared with different RNAs. For instance, we have comparative sequence analysis as one of the methods for RNA generation that can be shared by many RNA structures produced by the same method. The third table type is structure-based tables like bulges, hairpins, internal loops, multiloops, pseudoknots, and unpaired regions. The main table (RNA) has a one-to-many relationship with the structure-based tables. All of the relational information and details can be found in the database diagram. (Figure A.1)

The scripts for bpRNA-1m database creation and loading are available in MySQL format and can be downloaded from the bpRNA webserver. We also provide perl scripts to create and load the tables. Beside all the structural details in our database, we also have files like FASTA, BPSEQ, CT, dot-bracket, as well as other novel structure file formats, for each RNA that are accessible by the RNA-Name. Moreover, high-resolution structure drawings produced by VARNA(Darty, Denise, & Ponty, 2009) are provided, along with the structure representations.

In some structures, bpRNA automatically detected errors in the BPSEQ file. Table A.1 has the bpRNA ID, the original ID and the reason for changing the bpseq file. In some cases, a nucleotide X was pairing to nucleotide Y but also to 0, which means unpaired. In these cases, we selected the paired nucleotide. There were also some cases that the bases were paired to itself, which should not be possible.

A.4 Dot-bracket File

One of the outputs of bpRNA is a compact dot-bracket file. This file contains PK characters, or higher-order brackets such as "[", "{ ", and " < " , and does so with fewer symbols (lower page number) than RNA STRAND v2.0 (Figure A.2).

A.5 Structure Type File

One of the outputs of bpRNA is the structure type file. An example of this file for the structure in Figure 2.2 is presented as Figure A.3.

A.6 Annotation of Loops

As stated above, a segment can contain bulges and internal loops. In fact, all bulges and internal loops are within segments, and can readily be identified as unpaired nucleotides within a segment. Internal loops can be classified as either symmetric or asymmetric if the two strands have the same length or not. Bulges can be thought of as an asymmetric internal loop with one strand having zero length. These can be easily distinguished by the lengths of the unpaired strands.

As stated above, hairpin loops can be readily identified as self-edges of the segment graph. All other edges of the segment graph are either multiloop branches or external loop branches.

A.7 Identification of Multiloops and External Loops

While bulges and internal loops are easily identified, multiloop branches and external loops can be difficult to identify because they can have zero length due to some branches consisting of just a backbone. In order to distinguish multiloops and external loops, the segment graph needs to be identified (See Methods). After pseudoknot detection, and after the removal of base pairs involved in pseudoknots segments. At this point, all Hamiltonian cycles in the PK-free segment graph correspond to multiloops. In practice, we can make this easier by using adjacent base pairs to confirm the cycle. We create a branch graph, where the vertices are the putative multiloop or external loop branches, which should be all loops connecting segments. We form an edge between vertices if the loops are adjacent to a common base pair. The connected components of this graph, if any, are either multiloops or all the loops for the external loop. Each connected component that is a multiloop is such that each vertex is a branch of a multiloop.

A.8 Comparisons to other Databases

We found many differences between RNA STRAND v2.0, which is the most similar meta-database to what we have created in bpRNA-1m. In particular, when the nucleotides in a loop sequence are involved in a pseudoknot, these loops are no longer annotated as such. We take the approach that when a hairpin loop is involved in a pseudoknot, we still annotate it as a hairpin loop, but also label it as being involved in a PK.

We define bulges as a loop flanked by base pairs (i, j) and (i', j') where either $|i-i'| = 1$ or $|j - j'| = 1$. We suspect that although RNA STRAND v2.0 also uses this definition, there may be some condition on whether the loop is flanked by PK-forming nucleotides because we found 1,042 examples where our annotations differ regarding bulges. We couldn't directly compare the annotations because whereas we provide detailed annotation of each bulge, including flanking base pairs, start and end positions, loops sequence etc., RNA STRAND only reports the number of bulges.

One example is presented in Figures A.4. The secondary structure image from RNA STRAND v2.0 for *List.wels._AF440351_1-321* is presented in Figure A.4A. The RNA STRAND v2.0 annotation lists this structure as having 0 bulges. However, in Figure A.4 B we provide the bpRNA-1m structure, which has 4 bulges indicated with light green highlighting.

A.9 Secondary Structure Images

In addition to a plain secondary structure image, we provide structure images color-coded by different features extracted by bpRNA. First, is a structure image color-coded by segment number (Figure A.5A). Second, is a structure image color-coded by structure type (Figure A.5B). Third is a linear structure with base pairs color-coded by their page number (Figure A.5C).

A.10 Internal Loops

We observed some trends associated with the lengths of internal loops for various mismatch pairs. We found that while GA mismatches show a strong preference for internal loops of length 3, AA mismatches showed a strong preference for length 2 (Figure A.6).

A.11 Bulges

We further investigated the length distributions for bulges when separated by their closing base pairs. By far, the most prevalent bulges had GC closing base pairs (Figure A.7A). We found fewer bulges flanked by AU and GU pairs (Figure A.7B-C). We examined the length trends for different mismatch pairs when the bulge was greater than 1 nt. We found that bulges of length 2 nt were enriched AA mismatch pairs, and bulges of length 3 were enriched for GA mismatch pairs. We also found a strong enrichment for bulges of length 6 with GA mismatch pairs (Figure A.7D).

A.12 Multiloops

We examined the closing base pairs and branch lengths associated with multiloops of different branch-numbers (Figure A.8). We observed 4-branch loops showed a preference for C:G closing pairs (Figure A.8B) and that multiloop branches of length 0 for 4-branch loops had a strong preference for C:G pairs (Figure A.8E).

A.13 Pseudoknots

We found that 12% of structures in bpRNA-1m(90) have pseudoknots. For a detailed breakdown of the percentage of structures with PKs from each source database for both bpRNA-1m and bpRNA-1m(90), see Supplementary Table 2.

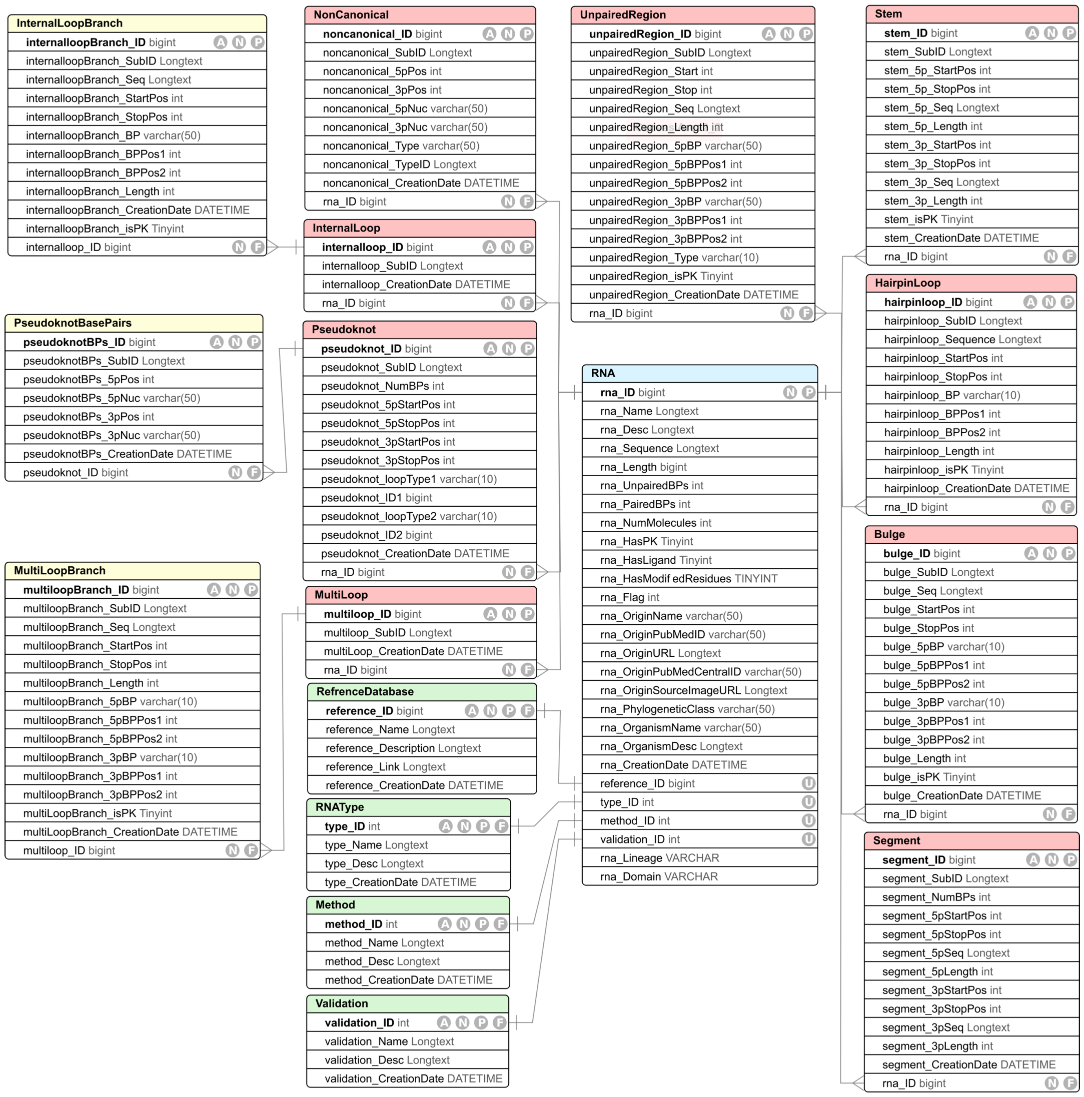

InternalLoopBranch
internalloopBranch_ID bigint
internalloopBranch_SubID Longtext
internalloopBranch_Seq Longtext
internalloopBranch_StartPos int
internalloopBranch_StopPos int
internalloopBranch_BP varchar(50)
internalloopBranch_BPPos1 int
internalloopBranch_BPPos2 int
internalloopBranch_Length int
internalloopBranch_CreationDate DATETIME
internalloopBranch_isPK Tinyint
internalloop_ID bigint
PseudoknotBasePairs
pseudoknotBPs_ID bigint
pseudoknotBPs_SubID Longtext
pseudoknotBPs_5pPos int
pseudoknotBPs_5pNuc varchar(50)
pseudoknotBPs_3pPos int
pseudoknotBPs_3pNuc varchar(50)
pseudoknotBPs_CreationDate DATETIME
pseudoknot_ID bigint
MultiLoopBranch
multiloopBranch_ID bigint
multiloopBranch_SubID Longtext
multiloopBranch_Seq Longtext
multiloopBranch_StartPos int
multiloopBranch_StopPos int
multiloopBranch_Length int
multiloopBranch_5pBP varchar(10)
multiloopBranch_5pBPPos1 int
multiloopBranch_5pBPPos2 int
multiloopBranch_3pBP varchar(10)
multiloopBranch_3pBPPos1 int
multiloopBranch_3pBPPos2 int
multiLoopBranch_isPK Tinyint
multiLoopBranch_CreationDate DATETIME
multiloop_ID bigint
NonCanonical
noncanonical_ID bigint
noncanonical_SubID Longtext
noncanonical_5pPos int
noncanonical_3pPos int
noncanonical_5pNuc varchar(50)
noncanonical_3pNuc varchar(50)
noncanonical_Type varchar(50)
noncanonical_TypeID Longtext
noncanonical_CreationDate DATETIME
rna_ID bigint
InternalLoop
internalloop_ID bigint
internalloop_SubID Longtext
internalloop_CreationDate DATETIME
rna_ID bigint
Pseudoknot
pseudoknot_ID bigint
pseudoknot_SubID Longtext
pseudoknot_NumBPs int
pseudoknot_5pStartPos int
pseudoknot_5pStopPos int
pseudoknot_3pStartPos int
pseudoknot_3pStopPos int
pseudoknot_loopType1 varchar(10)
pseudoknot_ID1 bigint
pseudoknot_loopType2 varchar(10)
pseudoknot_ID2 bigint
pseudoknot_CreationDate DATETIME
rna_ID bigint
MultiLoop
multiloop_ID bigint
multiloop_SubID Longtext
multiLoop_CreationDate DATETIME
rna_ID bigint
RefrenceDatabase
reference_ID bigint
reference_Name Longtext
reference_Description Longtext
reference_Link Longtext
reference_CreationDate DATETIME
RNAType
type_ID int
type_Name Longtext
type_Desc Longtext
type_CreationDate DATETIME
Method
method_ID int
method_Name Longtext
method_Desc Longtext
method_CreationDate DATETIME
Validation
validation_ID int
validation_Name Longtext
validation_Desc Longtext
validation_CreationDate DATETIME
UnpairedRegion
unpairedRegion_ID bigint
unpairedRegion_SubID Longtext
unpairedRegion_Start int
unpairedRegion_Stop int
unpairedRegion_Seq Longtext
unpairedRegion_Length int
unpairedRegion_5pBP varchar(50)
unpairedRegion_5pBPPos1 int
unpairedRegion_5pBPPos2 int
unpairedRegion_3pBP varchar(50)
unpairedRegion_3pBPPos1 int
unpairedRegion_3pBPPos2 int
unpairedRegion_Type varchar(10)
unpairedRegion_isPK Tinyint
unpairedRegion_CreationDate DATETIME
rna_ID bigint
RNA
rna_ID bigint
rna_Name Longtext
rna_Desc Longtext
rna_Sequence Longtext
rna_Length bigint
rna_UnpairedBPs int
rna_PairedBPs int
rna_NumMolecules int
rna_HasPK Tinyint
rna_HasLigand Tinyint
rna_HasModif edResidues TINYINT
rna_Flag int
rna_OriginName varchar(50)
rna_OriginPubMedID varchar(50)
rna_OriginURL Longtext
rna_OriginPubMedCentralID varchar(50)
rna_OriginSourceImageURL Longtext
rna_PhylogeneticClass varchar(50)
rna_OrganismName varchar(50)
rna_OrganismDesc Longtext
rna_CreationDate DATETIME
reference_ID bigint
type_ID int
method_ID int
validation_ID int
rna_Lineage VARCHAR
rna_Domain VARCHAR
Stem
stem_ID bigint
stem_SubID Longtext
stem_5p_StartPos int
stem_5p_StopPos int
stem_5p_Seq Longtext
stem_5p_Length int
stem_3p_StartPos int
stem_3p_StopPos int
stem_3p_Seq Longtext
stem_3p_Length int
stem_isPK Tinyint
stem_CreationDate DATETIME
rna_ID bigint
HairpinLoop
hairpinloop_ID bigint
hairpinloop_SubID Longtext
hairpinloop_Sequence Longtext
hairpinloop_StartPos int
hairpinloop_StopPos int
hairpinloop_BP varchar(10)
hairpinloop_BPPos1 int
hairpinloop_BPPos2 int
hairpinloop_Length int
hairpinloop_isPK Tinyint
hairpinloop_CreationDate DATETIME
rna_ID bigint
Bulge
bulge_ID bigint
bulge_SubID Longtext
bulge_Seq Longtext
bulge_StartPos int
bulge_StopPos int
bulge_5pBP varchar(10)
bulge_5pBPPos1 int
bulge_5pBPPos2 int
bulge_3pBP varchar(10)
bulge_3pBPPos1 int
bulge_3pBPPos2 int
bulge_Length int
bulge_isPK Tinyint
bulge_CreationDate DATETIME
rna_ID bigint
Segment
segment_ID bigint
segment_SubID Longtext
segment_NumBPs int
segment_5pStartPos int
segment_5pStopPos int
segment_5pSeq Longtext
segment_5pLength int
segment_3pStartPos int
segment_3pStopPos int
segment_3pSeq Longtext
segment_3pLength int
segment_CreationDate DATETIME
rna_ID bigint

Figure A.1: bpRNA-1m database diagram illustrates all the fields, fields type, and relational connection in the tables (created by SQLEditor). The primary keys of each tables are bold and has a "P" mark. If a field cannot be Null, is marked by "N" , the foreign keys are "F" s, and auto increments are "A" . The unique Ids are also shown by "U" . Each table has a creation date to keep track of the date/time that each data is added to the database.

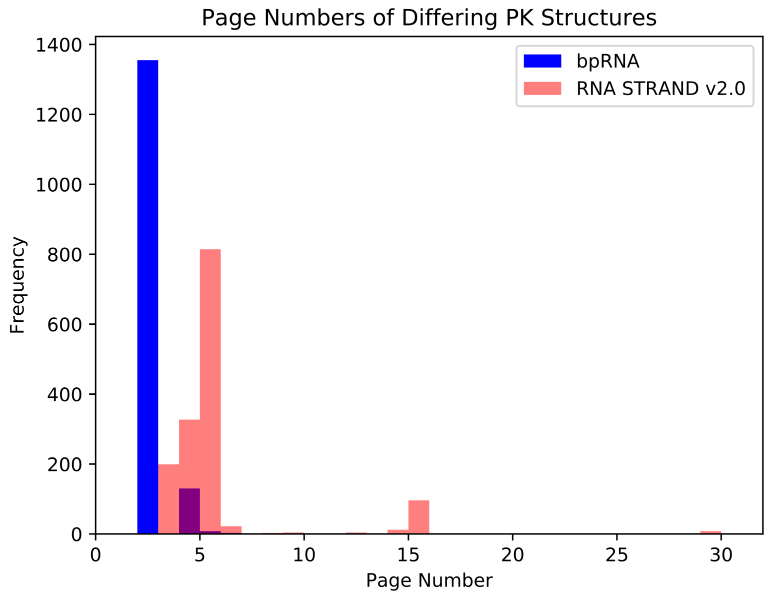
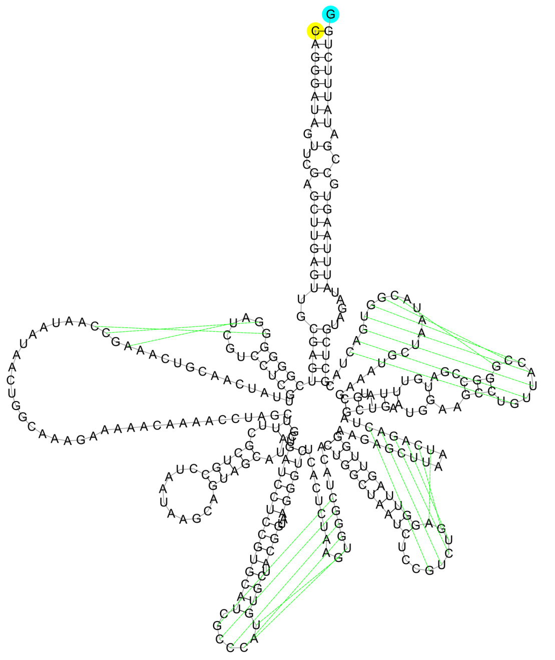


Figure A.2: Histogram compares the 1,497 structures where the page number differs for bpRNA-1m and RNA STRAND v2.0.

```
#Name: bpRNA_RFAM_34891
#Length:  191
AUCAGCCUUCGUUCUGUAAACGGGGUUGGAUCCGACUCUCAUAGGCUCUCCAACCCAACUCCUACUCAAUACGUCCU
CGUCGUACAGAACGGUAACAUGUUUUCCGAACAUCCGCGCUUGGGUAUACGAGUAUACACCUUACCCAACCCUCGCC
AACGGGGAGGAUGGAAAACAUGGCUAAAUUGAGAGGG
..........[[[[[[[.....((((((((.((..........))...))))))))...((((.(((((((((.{{.
...)))]]]]]]]}}..(((((((((((....((((....((((((.................)))))).(((....
....))).)))))))))))))))....))))))))))
EEEEEEEEEEEEEEEEEEEEEESSSSSSSSISSHHHHHHHHHHSSIIISSSSSSSSXXXSSSSBSSSSSSSSSHHHH
HHHSSSMMMMMMMMMMMSSSSSSSSSSSBBBBSSSSMMMMSSSSSSHHHHHHHHHHHHHHHHHSSSSSSMSSSHHHH
HHHHSSSMSSSSSSSSSSSSSSSMMMMSSSSSSSSSS
NNNNNNNNNNKKKKKKKNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNKKN
NNNNNNKKKKKKKKKNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
S1 23..30 "GGGUUGGA" 49..56 "UCCAACCC"
S2 32..33 "CC" 44..45 "GG"
S3 60..63 "UCCU" 188..191 "AGGG"
S4 65..70 "CUCAAU" 182..187 "AUUGAG"
S5 71..73 "ACG" 81..83 "CGU"
S6 95..105 "ACAUGUUUUCC" 167..177 "GGAAAACAUGG"
S7 110..113 "AUCC" 163..166 "GGAU"
S8 118..123 "UUGGGU" 141..146 "ACCCAA"
S9 148..150 "CCU" 159..161 "GGG"
H1 34..43 "GACUCUCAUA" C:G
H2 74..80 "UCCUCGU" G:C PK{2}
H3 124..140 "AUACGAGUAUACACCUU" U:A
H4 151..158 "CGCCAACG" U:G
B1 64..64 "A" U:A C:G
B2 106..109 "GAAC" C:G A:U
I1.1 31..31 "U" A:U
I1.2 46..48 "CUC" G:C
M1.1 71..70 "" U:A A:U
M1.2 84..94 "ACAGAACGGUA" U:A A:G PK{1,2}
M1.3 178..181 "CUAA" G:A A:U
M2.1 114..117 "GCGC" C:G U:A
M2.2 147..147 "C" A:U C:G
M2.3 162..162 "A" G:C G:C
X1 57..59 "AAC"
E1 1..22 "AUCAGCCUUCGUUCUGUAAACG" PK{1}
PK1 7bp 11..17 84..90 E1 1..22 M1.2 84..94
PK2 2bp 75..76 91..92 H2 74..80 M1.2 84..94
PK1.1 11 G 90 C
PK1.2 12 U 89 A
PK1.3 13 U 88 A
PK1.4 14 C 87 G
PK1.5 15 U 86 A
PK1.6 16 G 85 C
PK1.7 17 U 84 A
PK2.1 75 C 92 G
PK2.2 76 C 91 G
NCBP1 95 A 177 G S6
segment1 10bp 23..33 GGGUUGGAUCC 44..56 GGCUCUCCAACCC
segment2 10bp 60..70 UCCUACUCAAU 182..191 AUUGAGAGGG
segment3 3bp 71..73 ACG 81..83 CGU
segment4 15bp 95..113 ACAUGUUUUCCGAACAUCC 163..177 GGAUGGAAAACAUGG
segment5 6bp 118..123 UUGGGU 141..146 ACCCAA
segment6 3bp 148..150 CCU 159..161 GGG
```

Figure A.3: The "structure type" file for bpRNA record bpRNA_RFAM_34891, which corresponds to RFAM sequence RF01788_AAAB01008846.1_3701932-3701742, and shown as Figure 2.2. Note that unlike Figure 2.2, the pseudoknot segments and segments are renumbered here starting at 1.

A



B

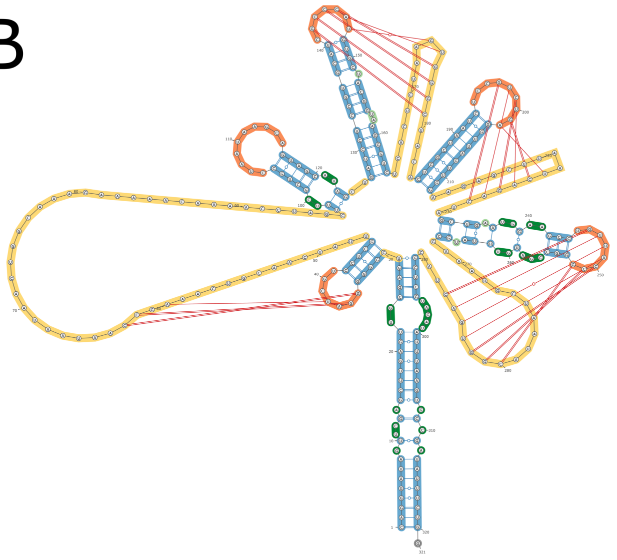
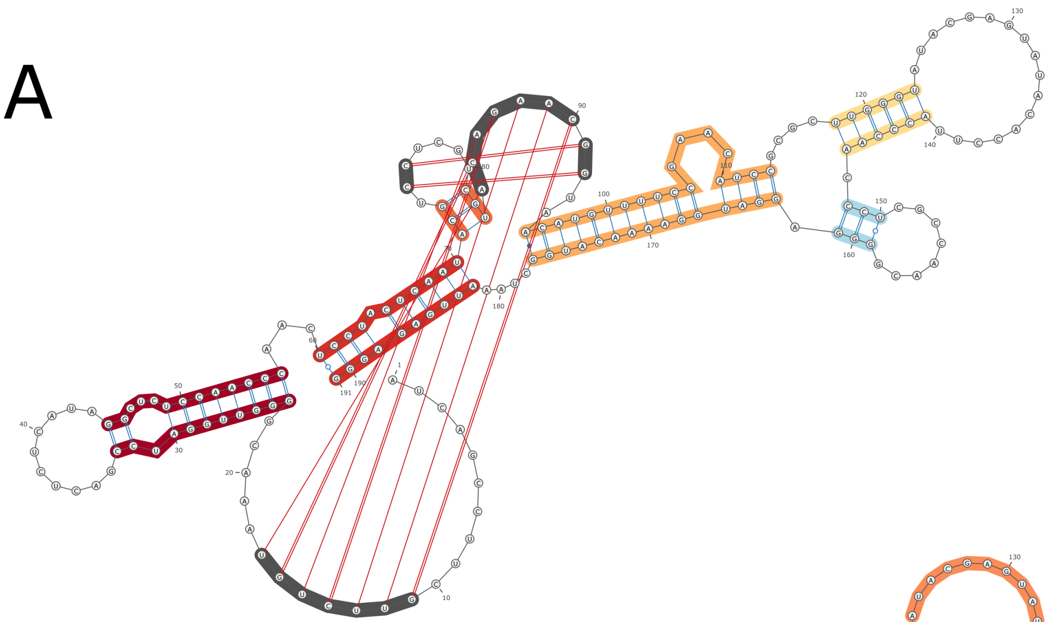
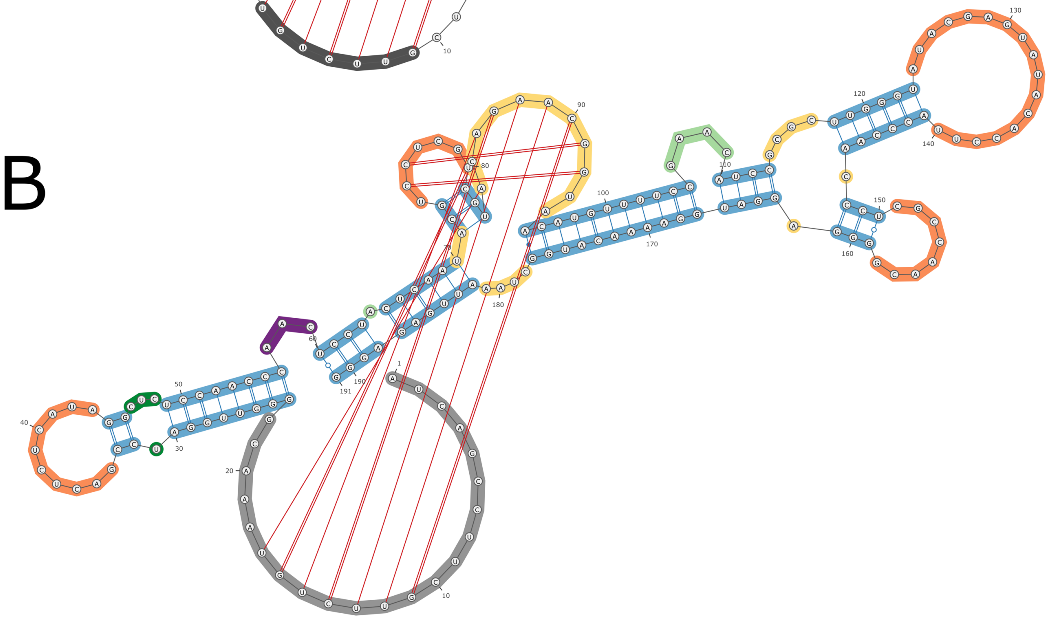


Figure A.4: The secondary structure image for record for List.wels._AF440351_1-321 (TMR_00327) in RNA STRAND v2.0. The structure annotation indicates that the structure has no bulges at time of writing (`http://www.rnasoft.ca/strand/show_results.php?molecule_ID=TMR_00327`. **B.** The bpRNA-1m secondary structure image has the same structure (apart from rotation), with the bulges indicated with light green.

A



B



C

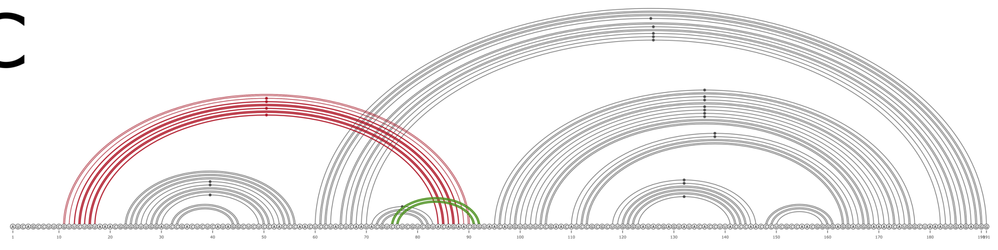
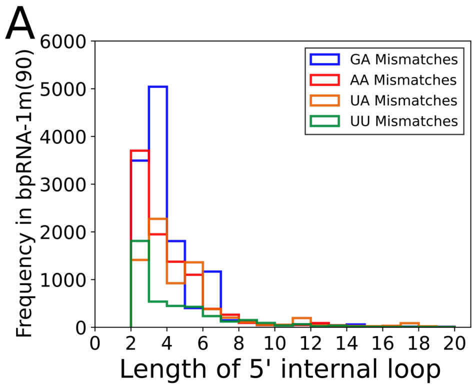


Figure A.5: Examples of the secondary structure images available for all structures in bpRNA-1m, for bpRNA_RFAM_34891, the same structure as in Figure 2. **A.** A secondary structure image color-coded by segment number **B.** A secondary structure image color-coded by structure type. **C.** A linear secondary structure image color-coded by page number.



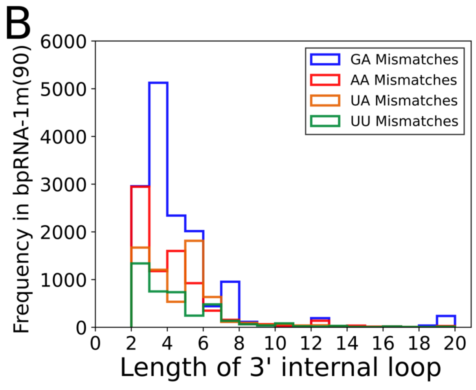


Figure A.6: **A.** The length distribution for the 5’ sequence of internal loops with the most common mismatch pairs. **B.** The length distribution for the 3’ sequence of internal loops with the most common mismatch pairs.

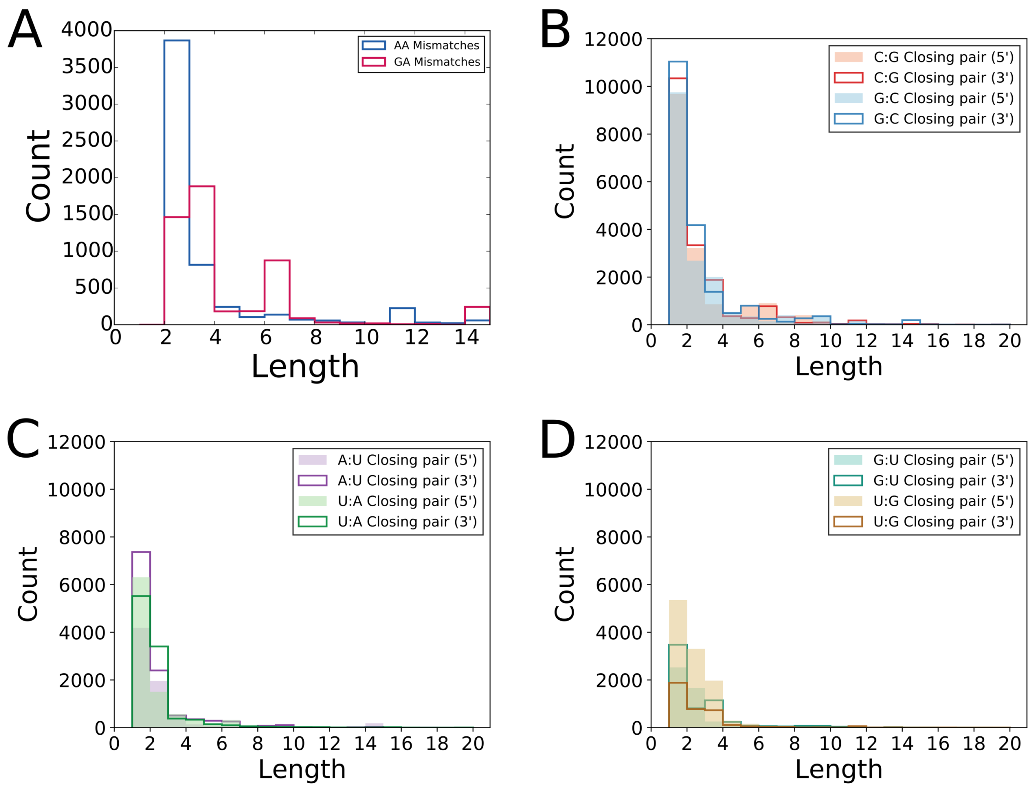


Figure A.7: **A.** The length distributions of bulges for GC closing base pairs. **B.** The length distributions for bulges with AC closing base pairs. **C.** The length distributions for bulges with GU closing pairs. **D.** The length distributions for bulges with the two most common mismatch pairs.

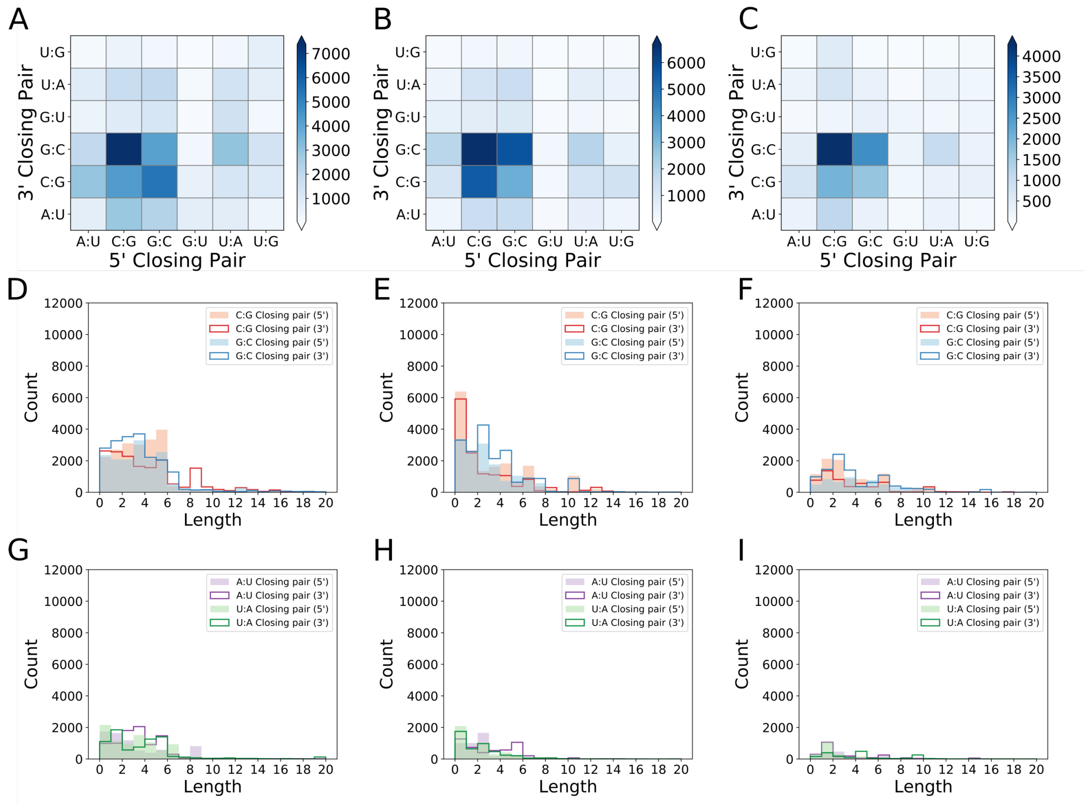


Figure A.8: **A.** The closing base pair frequency heat map for multiloop branches within 3-branch multiloops. **B.** The closing base pair frequency heat map for multiloop branches within 4-branch multiloops. **C.** The closing base pair frequency heat map for multiloop branches within 5-branch multiloops. **D.** The length distribution for multiloop branches within 3-branch multiloops for C:G and G:C closing base pairs. **E.** The length distribution for multiloop branches within 4-branch multiloops for C:G and G:C closing base pairs. **F.** The length distribution for multiloop branches within 5-branch multiloops for C:G and G:C closing base pairs. **G.** The length distribution for multiloop branches within 3-branch multiloops for A:U and U:A closing base pairs. **H.** The length distribution for multiloop branches within 4-branch multiloops for A:U and U:A closing base pairs. **I.** The length distribution for multiloop branches within 5-branch multiloops for A:U and U:A closing base pairs.

bpRNA ID	Original ID	Source	Reason for the change in bpseq File
bpRNA_CRW_54533	a.16.e.G.muris	CRW	positions 316 paired to 295 and 0
bpRNA_CRW_54853	d.16.b.H.aurantiacus	CRW	positions 702 paired to 610 and 0
bpRNA_CRW_55041	d.16.e.B.hominis	CRW	positions 585 paired to 27 and 0
bpRNA_CRW_54549	a.I1.b.Synechococcus.sp2.C3.tLEU	CRW	positions 203 paired to 150 and 1
bpRNA_CRW_54788	d.16.b.B.anthracis	CRW	positions 533 paired to 516 and 0
bpRNA_CRW_55390	d.235.c.E.gracilis	CRW	positions 2325 paired to 2283 and 0
bpRNA_CRW_54532	a.16.e.G.intraradices	CRW	positions 285 paired to 273 and 272
bpRNA_CRW_55315	d.233.c.E.gracilis	CRW	positions 2325 paired to 2283 and 0
bpRNA_CRW_55385	d.235.b.S.ambofaciens	CRW	positions 439 paired to 366 and 0
bpRNA_CRW_54715	b.16.m.M.polymorpha	CRW	positions 1248 paired to 1241 and 1240
bpRNA_CRW_54959	d.16.b.S.putrefaciens	CRW	positions 1525 paired to 1512 and 0
bpRNA_CRW_54689	b.16.e.C.koreana	CRW	positions 1076 paired to 1059 and 0
bpRNA_CRW_54531	a.16.e.G.ardeae	CRW	positions 318 paired to 297 and 0
bpRNA_CRW_55310	d.233.b.S.ambofaciens	CRW	positions 439 paired to 366 and 0
bpRNA_CRW_54616	a.I1.e.P.spiralis.D. C1.SSU.1506	CRW	positions 625 paired to 610 and 0
bpRNA_CRW_54653	a.I1.m.S.sclerotiorum.C2.SSU.570	CRW	positions 124 paired to 112 and 0
bpRNA_CRW_54615	a.I1.e.P.spiralis.B.C1.SSU.1506	CRW	positions 461 paired to 446 and 0
bpRNA_CRW_54904	d.16.b.P.marina	CRW	Position 47 is paired to both 339 and 1472
bpRNA_CRW_54853	d.16.b.H.aurantiacus	CRW	Position 611 is paired to both 700 and 701
bpRNA_CRW_55360	d.235.b.C.psittaci	CRW	Position 2852 is paired to both 2870 and 2924
bpRNA_CRW_54913	d.16.b.P.staleyi	CRW	Position 47 is paired to both 340 and 1525
bpRNA_CRW_54614	a.I1.e.P.sarcinoidea.C1.SSU.943	CRW	Position 413 is paired to both 439 and 441
bpRNA_CRW_55390	d.235.c.E.gracilis	CRW	Position 2283 is paired to 2325, but 2325 is paired to 0
bpRNA_SRP_317	Fuso.nucl._AE009951	SRP	Position 24 was paired to itself
bpRNA_SRP_852	Ther.teng._AE012978	SRP	Position 138 was paired to itself
bpRNA_SRP_582	Onch.volv._BF727619	SRP	Position 120 was paired to itself
bpRNA_SRP_581	Onch.volv._AI249203	SRP	Position 113 was paired to itself
bpRNA_RNP_426	N.olivacea-chloroplast	RNaseP	Position 18 is paired to both 40 and 45
bpRNA_RNP_434	P.purpurea-chloroplast	RNaseP	Position 20 is paired to both 42 and 47
bpRNA_RNP_413	C.paradoxa-cyanelle	RNaseP	Position 20 is paired to both 36 and 41

Table A.1: List of the RNA structures that needed modifications in their original bpseq files.

bpRNA-1m	**PKs**	**Total**	**Percentage**
CRW	16,668	55,600	29.98
tmRNA	713	728	97.94
RNP	381	466	81.76
RFAM	3,941	43,273	9.11
PDB	238	669	35.58
SRP	0	959	0.00
tRNAdb	0	623	0.00
Total	21,941	102,318	21.44
bpRNA-1m(90)	**PKs**	**Total**	**Percentage**
CRW	1,095	4,368	25.07
tmRNA	332	339	97.94
RNP	189	253	74.70
RFAM	1,642	22,521	7.29
PDB	62	330	18.79
SRP	0	352	0.00
tRNAdb	0	207	0.00
Total	3,320	28,370	11.70

Table A.2: The percentage of structures with PKs from each source database.

Appendix B: Model Interpretability as Part of a Rule-based method for Fast and Accurate RNA Pseudoknot Detection

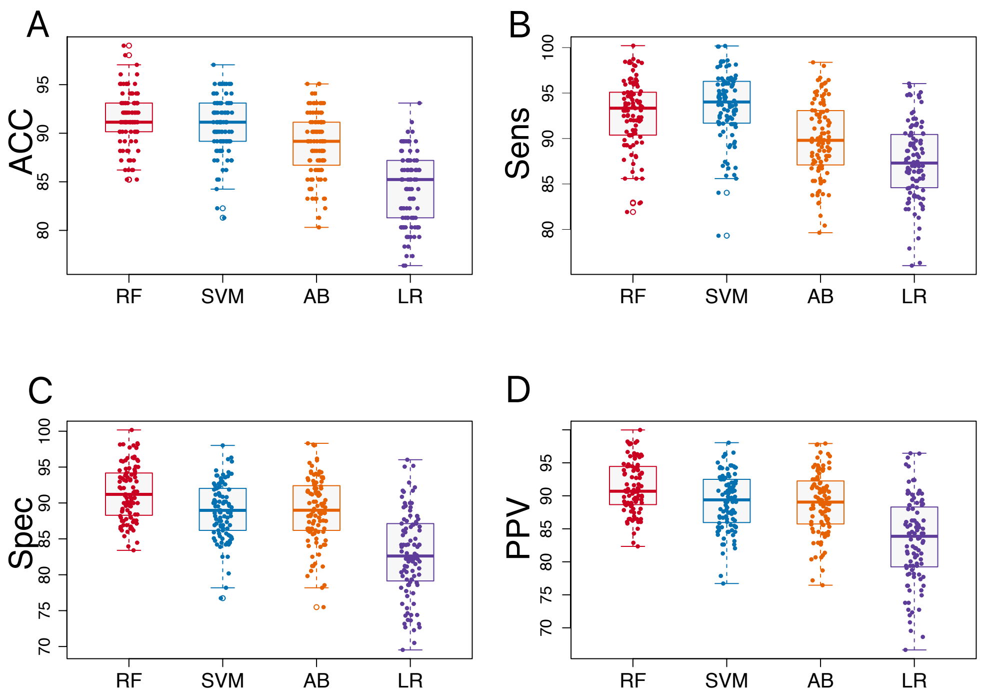


Figure B.1: Box plots of accuracy, sensitivity, specificity, and positive predicted value using LPO cross-validation on CompaRNA. Similar to what we had in Figure 3.3, we captured the results from 100 iterations using four different ML models on the original CompaRNA set. Likewise, RF and SVM have the highest value on average in comparison with AB and LR models.

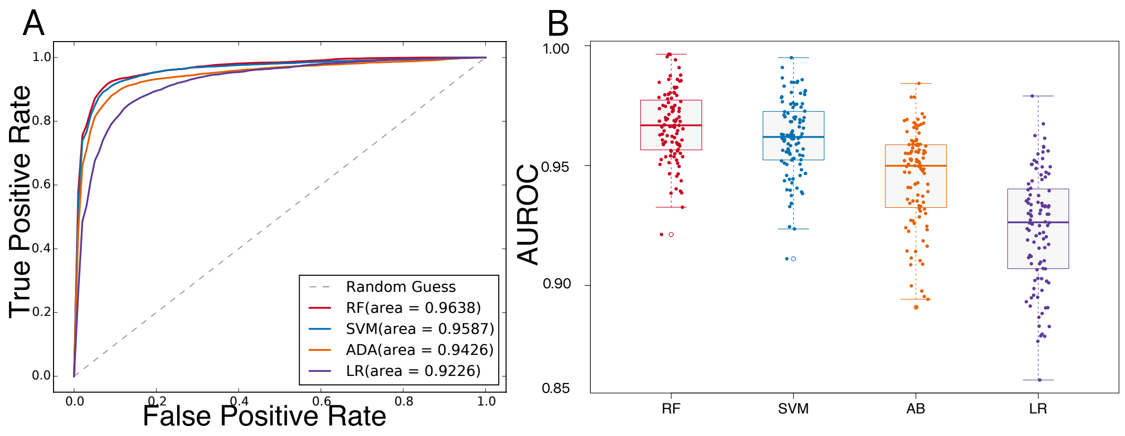


Figure B.2: ROC curve and AUROC box plots of four different machine learning techniques on CompaRNA. **A.** Represents the ROC curve comparison of our ML models. The RF and SVM models have the highest AUROC values on average (96.38 % and 98.57 % respectively) **B.** The box plots of AUROC values in 100 iterations of LPO cross-validation also represents that RF and SVM perform best using the original CompaRNA set.

Table B.1: Pseudoknow-assisted folding median results using CompaRNA-1m. Similar to what we had in Table 3.3, `Pseudoknow` performs better than the average of the two structure prediction algorithms (`IPknot` and `ContraFold`). While the median ACC was slightly better for `CONTRAfold`, our Pseudoknow-assisted folding has a median ACC that is higher than the average of `IPknot` and `CONTRAfold`, and unlike `CONTRAfold`, our strategy predicts pseudoknots.

Software	**ACC(%)**	**Sens(%)**	**Spec(%)**	**PPV(%)**	**MCC**
`Pseudoknow`	99.76	60.19	99.95	80.86	0.694
`IPknot`	99.72	50.00	99.95	78.16	0.625
`Contrafold`	99.77	68.67	99.93	77.94	0.729

Table B.2: Pseudoknow-assisted folding median results using PDB-validation. Similar to what we had in Table 3.4, `Pseudoknow` performs better than the average of the two structure prediction algorithms (`IPknot` and `ContraFold`

Model	**ACC(%)**	**Sens(%)**	**Spec(%)**	**PPV(%)**	**MCC**	**TP**	**FN**	**TN**	**FP**
`Pseudoknow`	79.37	75.00	80.85	57.14	0.516	12	4	38	9
`IPknot`	77.78	31.25	93.62	62.50	0.325	5	11	44	3
`ProbKnot`	77.78	18.75	97.87	75.00	0.297	3	13	46	1
`DotKnot`	85.71	68.75	91.49	73.33	0.616	11	5	43	4

Table B.3: Average PK-detection accuracy for `Pseudoknow` and other software on each specific subset of CompaRNA-1m.

Database	**PK**	**PKF**	**Pseudoknow**	**IPknot**	**ProbKnot**	**DotKnot**	**McQFold**	**pknotsRG**
ASE	218	67	75.33	69.12	45.61	78.94	65.96	29.82
CRW	331	115	90.84	81.39	77.57	76.91	35.87	51.12
PDB	25	159	92.52	87.50	85.87	89.13	90.76	90.76
RFA	16	117	83.51	72.18	77.44	59.40	91.72	80.45
NDB	0	1	100	100	0	0	100	100
SRP	0	244	97.84	61.88	83.19	57.38	52.45	80.74
SPR	0	230	100	62.60	74.78	39.13	94.78	94.78
TMR	235	2	93.67	78.90	54.43	82.27	70.88	13.92
Average	-	-	91.71	76.70	62.36	60.40	75.30	67.70

For further evaluation, we divided the RNAs into different subsets based on the database that they belong to. We set aside each subset as a validation and trained on the rest. The accuracy results demonstrate that in almost every subset, `Pseudoknow` performs better. In addition, we outperform on average compared with other RNA folding algorithms. The subsets correspond to the database source of each structure, RNase P Database (ASE), Gutell Lab CRW Site (CRW), RCSB Protein Data Bank (PDB),Rfam Database (RFA), Nucleic Acid Database (NDB), SRP Database (SRP), Sprinzl tRNA Database (SPR), tmRNA Database (TMR).

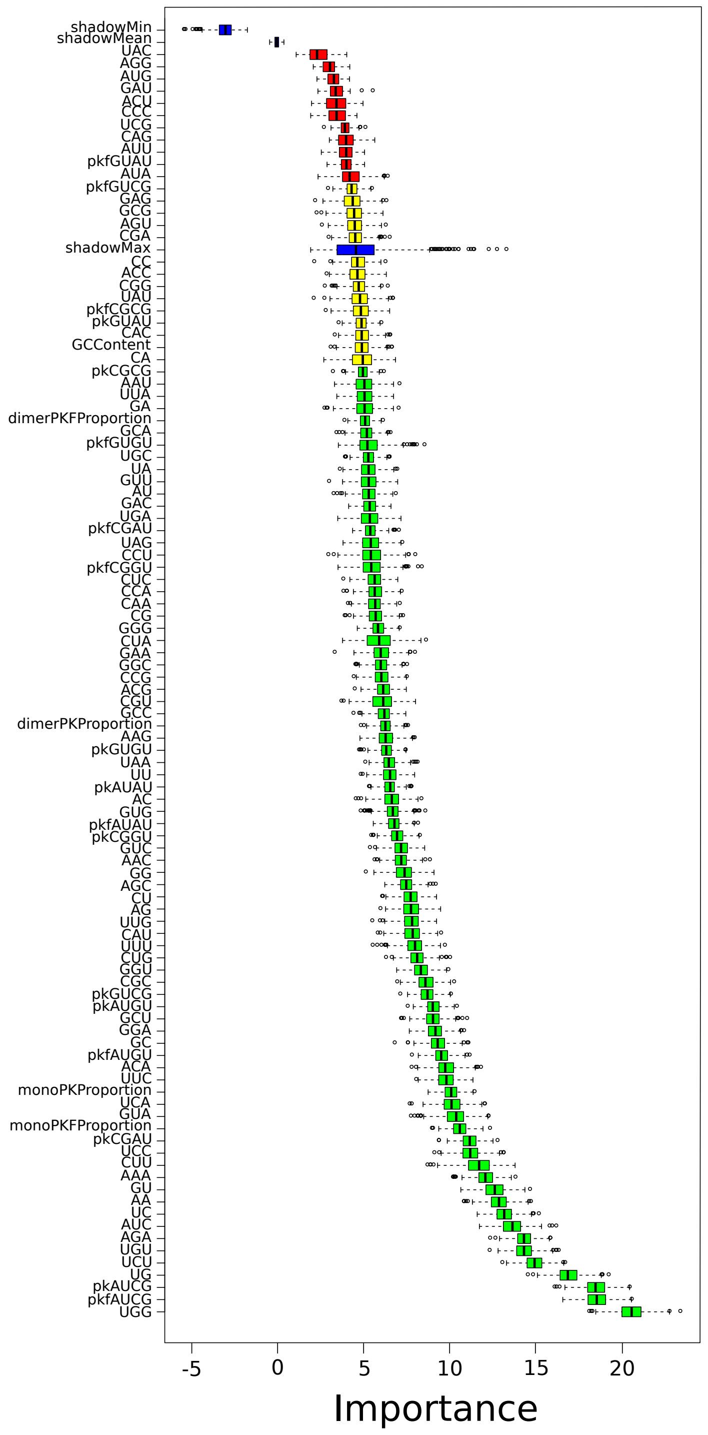


Figure B.3: Feature Importance in `Pseudoknow` using Brouta package. The importance of each feature is calculated by using the Random Forest classification algorithm with wrapper (Boruta feature selection). Green features represent the most powerful ones in our classification problem.

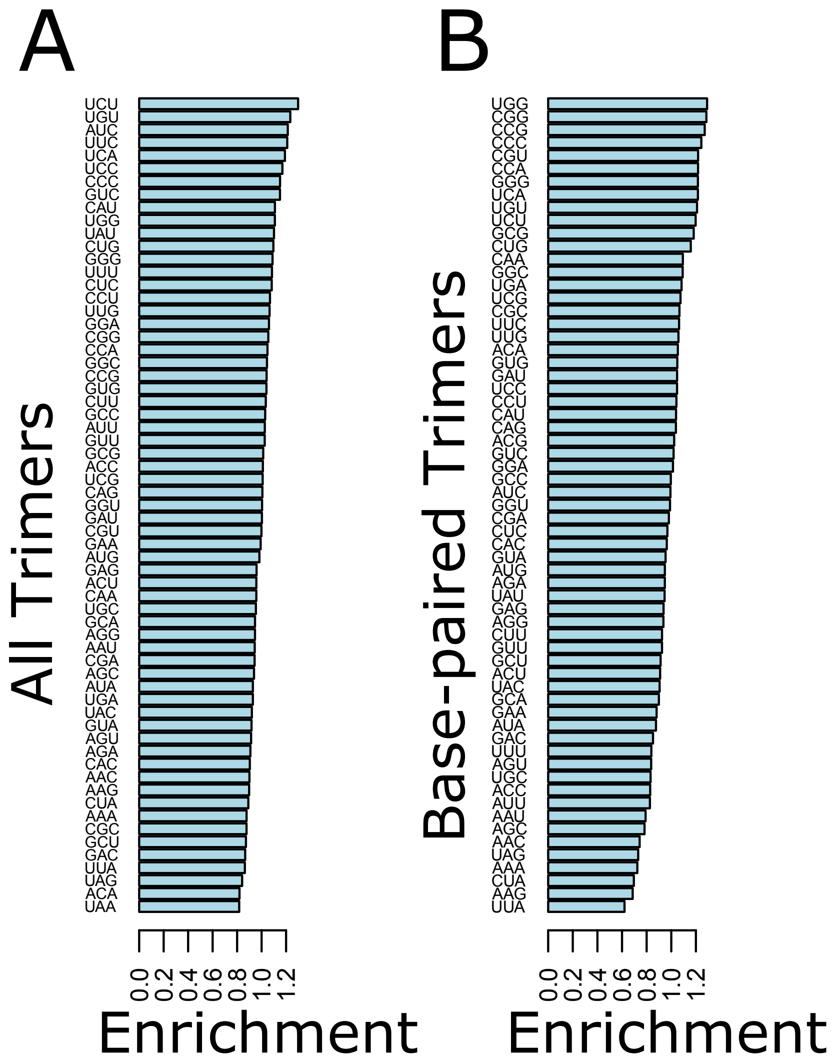


Figure B.4: Enrichment of trimers in PK versus PKF sequences and structures. Bar graph showing the enrichment of trimers in total **A.** and those participating in base pairs **B.** in PKF over PK.

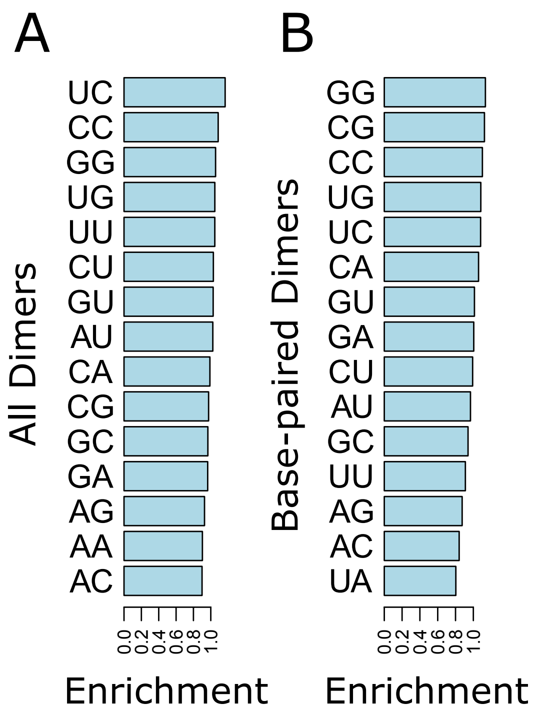


Figure B.5: Enrichment of dimers in PK versus PKF sequences and structures. Bar graph showing the enrichment of dimers in total **A.** and those participating in base pairs **B.** in PKF over PK.

Appendix C: An Interpretable Deep Learning Approach for Cancer Detection and Relevant Gene Identification

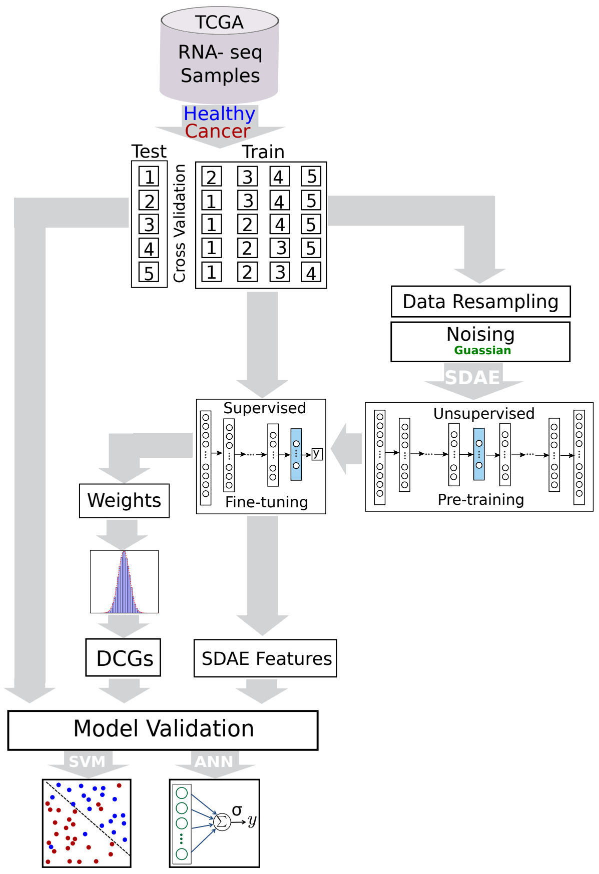


Figure C.1: The pipeline representing the stacked denoising autoencoder (SDAE) model for breast cancer classification and the process of biomarkers extraction.

GO biological process	Total	Observed	Expected	Enrichment	P-value
cell cycle process (GO:0022402)	1079	100	16.46	6.07	1.12E-45
cell cycle (GO:0007049)	1311	108	20	5.4	3.28E-45
mitotic cell cycle process (GO:1903047)	741	85	11.31	7.52	1.06E-44
mitotic cell cycle (GO:0000278)	760	85	11.6	7.33	7.33E-44
nuclear division (GO:0000280)	470	63	7.17	8.78	1.52E-35
organelle fission (GO:0048285)	492	64	7.51	8.53	1.99E-35
mitotic nuclear division (GO:0007067)	357	56	5.45	10.28	1.34E-34
cell division (GO:0051301)	477	58	7.28	7.97	3.72E-30
chromosome segregation (GO:0007059)	274	46	4.18	11	5.46E-29
sister chromatid segregation (GO:0000819)	176	36	2.69	13.41	7.62E-25
nuclear chromosome segregation (GO:0098813)	230	38	3.51	10.83	3.97E-23
mitotic cell cycle phase transition (GO:0044772)	249	35	3.8	9.21	8.10E-19
mitotic prometaphase (GO:0000236)	99	25	1.51	16.55	1.56E-18
cell cycle phase transition (GO:0044770)	255	35	3.89	9	1.72E-18
regulation of cell cycle (GO:0051726)	943	62	14.39	4.31	2.48E-18
chromosome organization (GO:0051276)	984	63	15.01	4.2	4.15E-18
DNA metabolic process (GO:0006259)	768	52	11.72	4.44	2.67E-15
organelle organization (GO:0006996)	3133	112	47.8	2.34	4.27E-15
mitotic cell cycle phase (GO:0098763)	211	29	3.22	9.01	7.67E-15
cell cycle phase (GO:0022403)	211	29	3.22	9.01	7.67E-15
biological phase (GO:0044848)	215	29	3.28	8.84	1.25E-14
sister chromatid cohesion (GO:0007062)	113	22	1.72	12.76	1.18E-13
cellular resp. to DNA damage stimu. (GO:0006974)	719	48	10.97	4.38	1.27E-13
regulation of cell cycle process (GO:0010564)	557	42	8.5	4.94	2.53E-13
mitotic sister chromatid segregation (GO:0000070)	90	20	1.37	14.56	3.01E-13
cell cycle checkpoint (GO:0000075)	196	25	2.99	8.36	1.09E-11
M phase (GO:0000279)	173	23	2.64	8.71	6.55E-11
mitotic M phase (GO:0000087)	173	23	2.64	8.71	6.55E-11
regulation of mitotic cell cycle (GO:0007346)	461	35	7.03	4.98	1.10E-10
single-organism process (GO:0044699)	12451	253	189.98	1.33	4.67E-10
DNA replication (GO:0006260)	213	24	3.25	7.38	5.74E-10
anaphase (GO:0051322)	154	21	2.35	8.94	6.13E-10
mitotic anaphase (GO:0000090)	154	21	2.35	8.94	6.13E-10
cellular component organization (GO:0016043)	5133	139	78.32	1.77	7.74E-10

Table C.1: Enriched GO terms associated with DCGs in breast cancer data from TCGA.

Appendix D: Interpretation of the Rules Learned by a Deep Convolutional Neural Network to Recognize Core Promoters

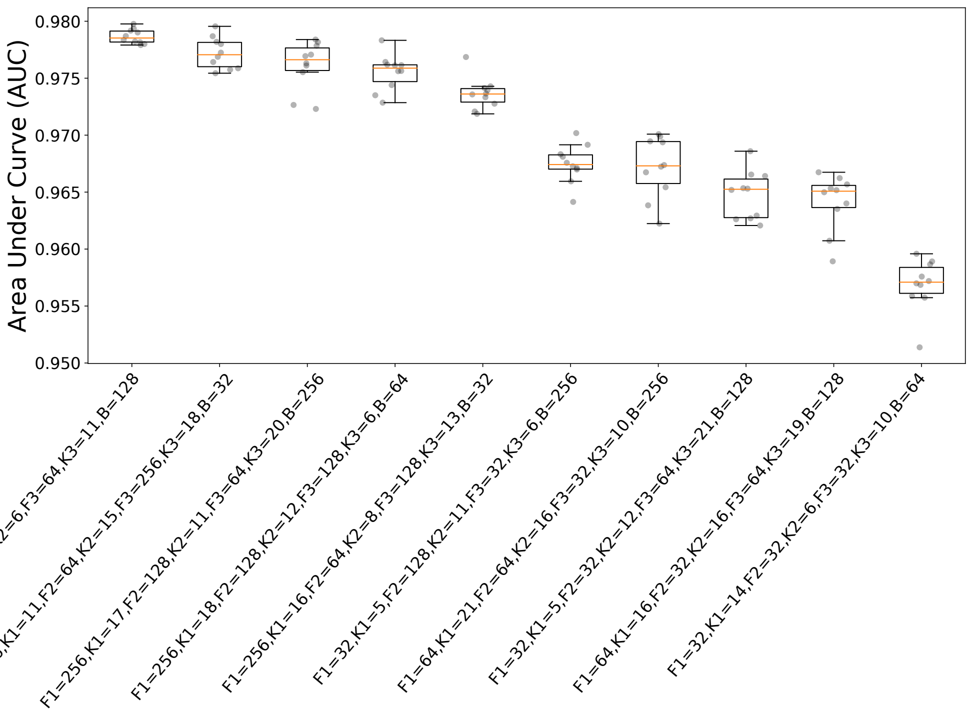


Figure D.1: Tuning plot of the top 10 CNN models for TSS detection using hyperas package.

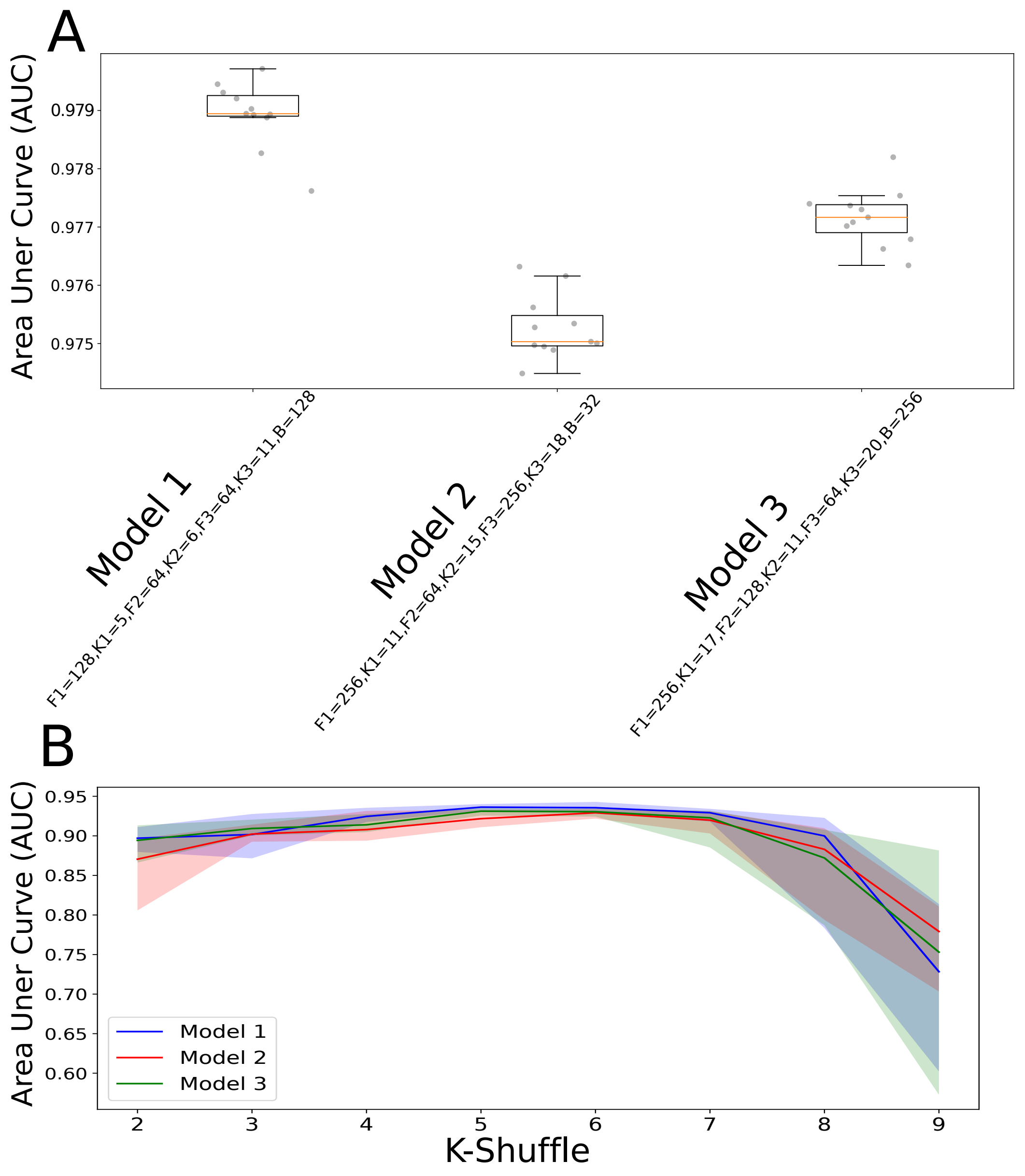


Figure D.2: Tuning plots of top 3 selected models are represented here. **A.** The box plot of top 3 selected models over 10 replicates validates that model 1 has the highest performance on the validation set compared to model 2 and 3. **B.** Additional k-shuffled tuning for the top 3 models depicts that $k = 6$ is still a better parameter value.

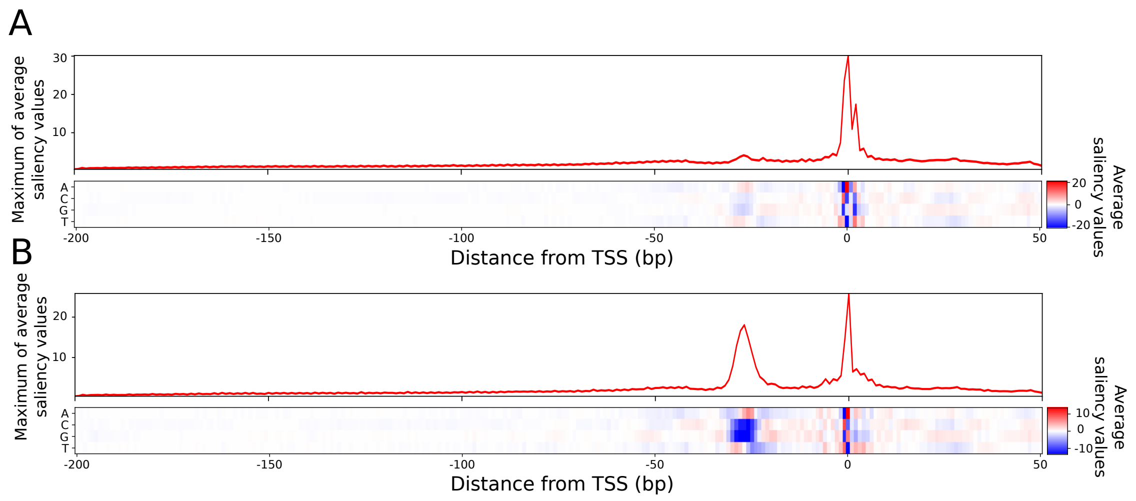


Figure D.3: Average Saliency maps representations over EPDnew data. **A.** Heatmap of average saliency map values using initiator sequences only. **B.** The heatmap representation of average saliency maps over TATA-containing sequences.

Table D.1: Initiator filters identified by our deep learning model using CAGE data. This table demonstrates that the CNN network was able to capture the variability between initiator motifs, which together assist in accurately detecting core promoter sequences.

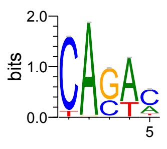
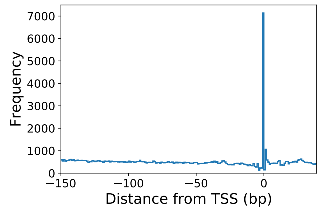
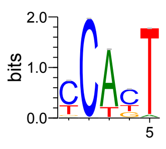
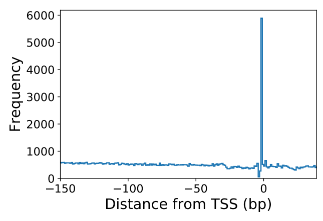
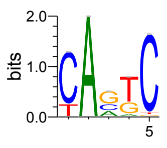
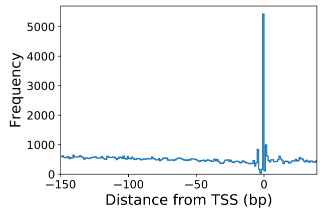
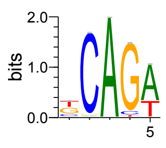
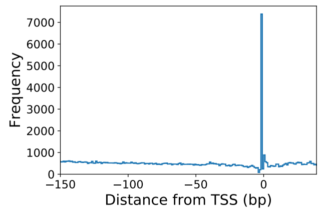
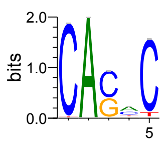
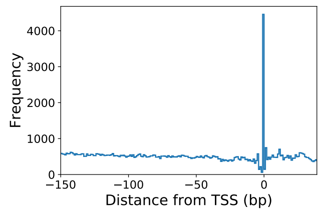
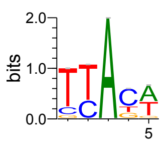
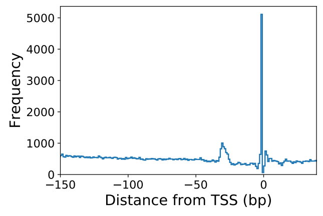
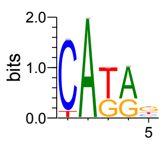
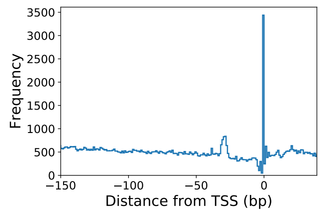
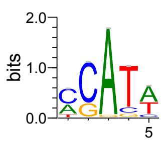
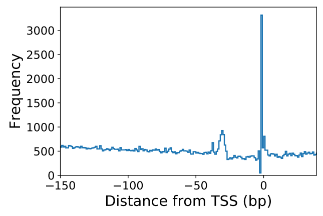
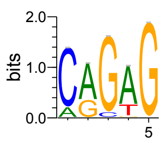
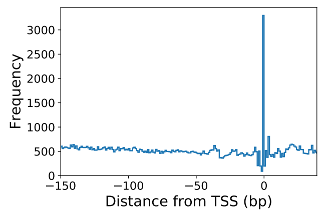
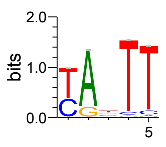
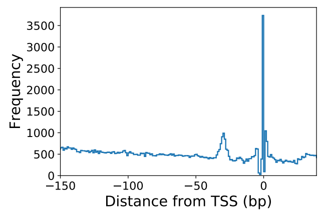
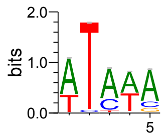
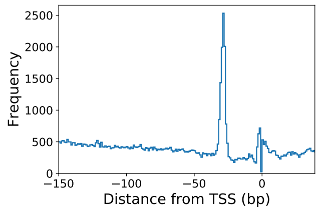
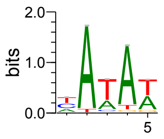
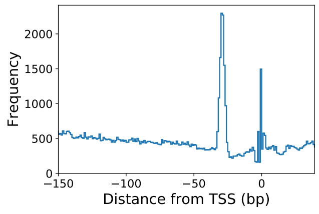
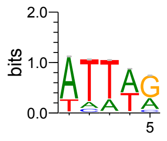
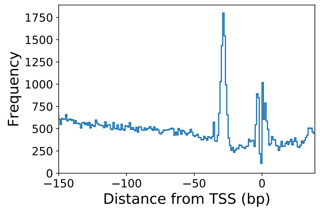
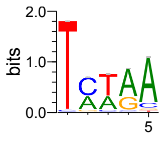
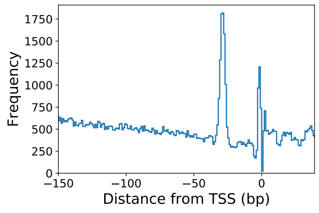
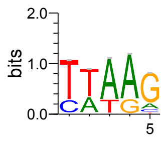
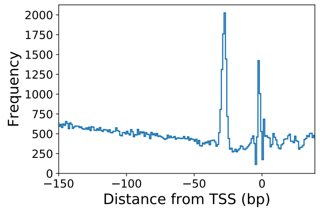
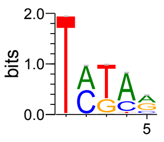
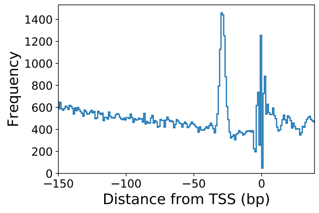
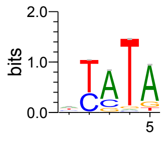
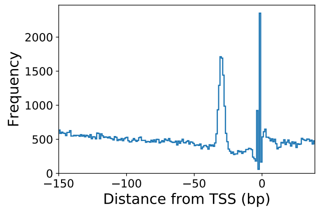
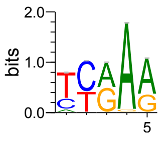
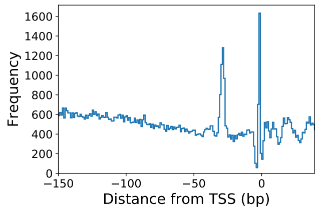
Filter	Logo	Position histogram		Filter	Logo	Position histogram
94				5		
49				32		
60				84		
127				47		
69				58		

Table D.2: TATA-like filters are represented in this table along with their logo and positional histogram using the CAGE dataset. This table represents that our CNN model could identify TATA motifs from core promoter sequences.

Filter	Logo	Position histogram		Filter	Logo	Position histogram
3				122		
31				57		
66				10		
46				120		

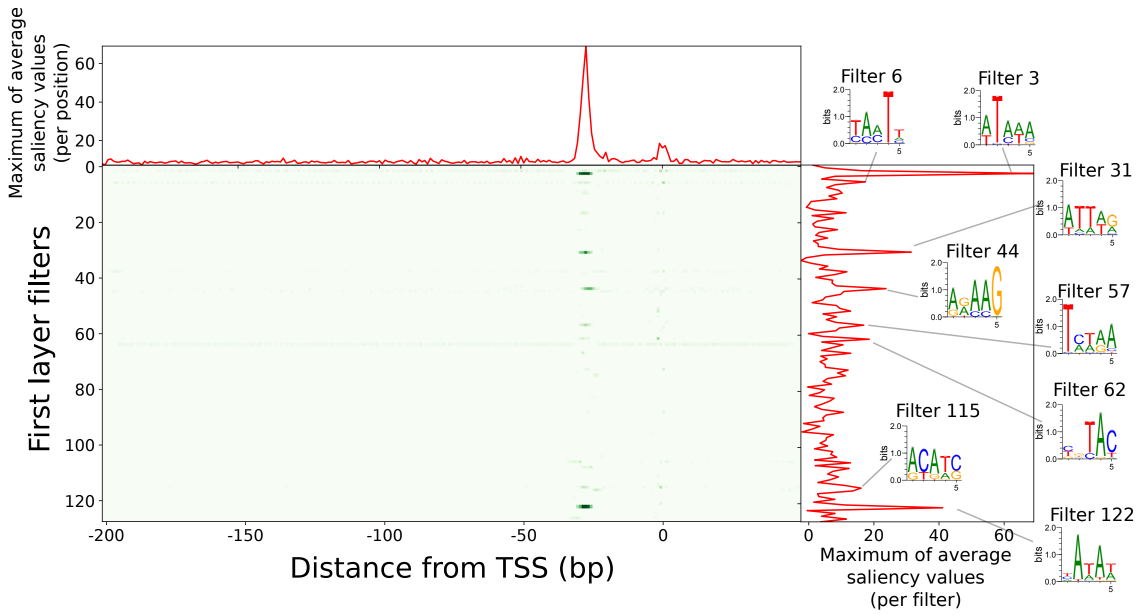


Figure D.4: The saliency maps of layer 2 convolutional neural network on the first layer convolutional network in TATA-containing sequences. This plot represents the effect of second layer on the first layer using the saliency map techniques averaged over all TATA-containing sequences that were extracted from EPD database.