



ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ
ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
“ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ & ΥΠΗΡΕΣΙΕΣ”

Non-Coding RNA Classifier
by
Konstantinos Vasilas

Submitted
in partial fulfilment of the requirements for the degree of
Master of Information Systems & Services: Big data and Analytics
at the
UNIVERSITY OF PIRAEUS
February 2024

Thesis Supervisor: Ilias Maglogiannis
Title: Dean of School

University of Piraeus,. All rights reserved.
Author Konstantinos Vasilas

ΣΕΛΙΔΑ ΕΓΚΥΡΟΤΗΤΑΣ

Όνοματεπώνυμο Φοιτητή/Φοιτήτριας: Κωνσταντίνος Βάσιλας

Τίτλος Μεταπτυχιακής Διπλωματικής Εργασίας: Non Coding RNA Classifier

Η παρούσα Μεταπτυχιακή Διπλωματική Εργασία υποβάλλεται ως μερική εκπλήρωση των απαιτήσεων του Προγράμματος Μεταπτυχιακών Σπουδών "Πληροφοριακά Συστήματα & Υπηρεσίες" του Τμήματος Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς και εγκρίθηκε στις 28/02/2024 από τα μέλη της Εξεταστικής Επιτροπής.

Εξεταστική Επιτροπή

Επιβλέπων (Τμήμα Ψηφιακών Συστημάτων, Πανεπιστήμιο Πειραιώς):
Ηλίας Μαγκλογιάννης, Κοσμήτορας Σχολής

Μέλος Εξεταστικής Επιτροπής: **Κωνσταντίνος Δελιμπάσης, Καθηγητής**

Μέλος Εξεταστικής Επιτροπής: **Δημοσθένης Κυριαζής, Καθηγητής**

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ ΑΥΘΕΝΤΙΚΟΤΗΤΑΣ

Ο Κωνσταντίνος Βάσιλας γνωρίζοντας τις συνέπειες της λογοκλοπής, δηλώνω υπεύθυνα ότι η παρούσα εργασία με τίτλο «Non Coding RNA Classifier», αποτελεί προϊόν αυστηρά προσωπικής εργασίας και όλες οι πηγές που έχω χρησιμοποιήσει, έχουν δηλωθεί κατάλληλα στις βιβλιογραφικές παραπομπές και αναφορές. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο ή/και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή.

Επιπλέον δηλώνω υπεύθυνα ότι η συγκεκριμένη Μεταπτυχιακή Διπλωματική Εργασία έχει συγγραφεί από εμένα προσωπικά και δεν έχει υποβληθεί ούτε έχει αξιολογηθεί στο πλαίσιο κάποιου άλλου μεταπτυχιακού ή προπτυχιακού τίτλου σπουδών, στην Ελλάδα ή στο εξωτερικό.

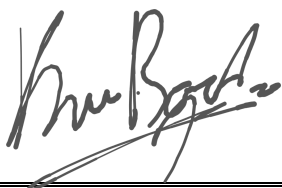
Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου. Σε κάθε περίπτωση, αναληθούς ή ανακριβούς δηλώσεως, υπόκειμαι στις συνέπειες που προβλέπονται τις διατάξεις που προβλέπει η Ελληνική και Κοινοτική Νομοθεσία περί πνευματικής ιδιοκτησίας.

Ο ΔΗΛΩΝ

Όνοματεπώνυμο: Κωνσταντίνος Βάσιλας

Αριθμός Μητρώου: ME2102

Υπογραφή:





UNIVERSITY OF PIRAEUS

Non Coding RNA Classifier

Author:

Konstantinos Vasilas
Student ID: ME2102
vasilas.cei@gmail.com

Supervisor:

Ilias Maglogianis

February 22, 2024

Abstract

This thesis presents a simplified yet effective deep learning model for the classification of non-coding RNAs (ncRNAs). Non-coding RNAs play a vital role in gene regulation and are associated with various biological processes and diseases. The complexity and diversity of ncRNAs make their classification a challenging task. To tackle this, a new neural network model called NCC was developed specifically designed to recognize patterns in ncRNA sequences as well as an updated collection of non-coding RNA sequences datasets to train and test the proposed architecture. The NCC model's performance, when benchmarked against traditional classifiers and existing RNA tools, revealed a 6% improvement in accuracy over the previously best-performing models, reaching 92.69% accuracy, along with slight enhancements in reliability, while still retaining its uncomplicated architecture. This model was trained and evaluated using a newly developed dataset that is ten times larger than the conventional dataset, achieving an accuracy rate exceeding 98%. The model's accuracy and interpretability hold potential for future research in genomic analysis and the identification of novel ncRNAs.

Contents

List of Figures

List of Tables

1	Introduction	1
1.1	Document Structure	2
2	RNA - Theoretical Background	3
2.1	RNA	3
2.2	Non-coding RNA	3
2.3	RNA Secondary and Tertiary Structure	5
3	Artificial Intelligence	7
3.1	Machine Learning	7
3.2	Classification Algorithms	8
3.3	Artificial Neural Networks	8
3.3.1	Neuron	9
3.4	Activation Functions	10
3.5	Deep learning	12
3.6	Fully Connected layer	12
3.7	Convolutional Neural Network (CNN)	12
3.7.1	Convolution Layer	13
3.7.2	Pooling Layer	13
3.8	Recurrent Neural Network (RNN)	14
3.8.1	Long-Short term memory (LSTM)	16
3.8.2	Gated Recurrent Unit (GRU)	17
3.8.3	Bidirectional RRN	17
4	Technologies	19
4.1	Python	19
4.2	Deep learning Frameworks	19
4.2.1	Tensorflow	19
4.2.2	Keras	19
4.2.3	PyTorch	19
4.2.4	Key Differences	20
4.3	Other Python Libraries	20
4.3.1	NumPy	20
4.3.2	Pandas	21
4.3.3	Scikit-learn	21
4.3.4	Matplotlib	21
4.4	RNA Secondary Structure Predict	21
4.4.1	Knotty	21
4.4.2	IPknot	22
4.4.3	Knotify	22

5	Related Work	24
5.1	RNAcon	24
5.2	nRC	25
5.3	GraPPLE	25
5.4	ncRFP	26
5.5	ncDLRES	26
5.6	ncDENSE	27
6	ncRNA data	28
6.1	ncRNA databases	28
6.2	RNACentral	29
6.3	IRESbase	30
6.4	Rfam	30
6.4.1	Rfam MySQL database	31
6.5	Data Collection	31
6.5.1	Fasta files	33
6.6	Final dataset (NCC dataset)	34
6.7	nRC dataset	36
6.7.1	Comparison between NCC and nRC datasets	37
7	Implementing the Prediction Mechanism.	39
7.1	Data preparation	39
7.1.1	Sequence Padding and Cutting	39
7.1.2	One-hot encoding	40
7.2	NCC Model Architecture	42
7.3	Training and Testing the Model	44
7.3.1	Performance metrics	45
7.3.2	Benchmarking NCC dataset	49
7.3.3	Benchmarking nRC dataset	50
8	Conclusion	52
9	Future Work	53
	Bibliography	55

List of Figures

1	Non-coding RNA Classes and sub-Classes	4
2	RNA Secondary and Tertiary structure (Image source [1])	5
3	Diagram of perceptron neuron	10
4	Fully connected (a) and convolution (b) layer example	14
5	RNN node Unfolding	15
6	LSTM cell Architecture	16
7	GRU cell Architecture	18
8	Layer of a Bidirectional RRN	18
9	Knotify approach representation (Image Source [2])	23
10	RNAcon Algorithm - a non-coding RNA Classifier [Image Source]	24
11	Pipeline of the nRC ncRNA sequence classification tool	25
12	ncRFP NN model Architecture (Image Source [3])	26
13	RNAcentral Expert Databases [Image source]	29
14	Rfam core database scheme diagram [Image Source]	32
15	Number of sequences per class distribution of final dataset (NCC dataset)	35
16	Sequence length distribution of miRNA and ribozyme	36
17	Sequence length mean and STB, per class, sorted by STB of NCC dataset	36
18	Sequence length mean and STB, per class, sorted by STB of nRC dataset	37
19	Sequence length distribution of leader RNA class in nRC (a) and NCC (b) dataset	38
20	Sequence length distribution of tRNA RNA class in nRC (a) and NCC (b) dataset	38
21	Sequence length distribution of ribozyme RNA class in nRC (a) and NCC (b) dataset	38
22	NCC dataset, RNA Sequences length distribution	40
23	NCC Neural Network Architecture	43
24	Diagram of Accuracy and loss per epoch during training	45
25	Confusion Matrix for binary classification problem	47
26	Confusion Matrix for multi-class classification problem	48
27	Confusion Matrix of NCC model trained and tested with NCC dataset matrix (a) is with 8 digits encoding and matrix (b) with 4 digits per RNA base	49
28	Confusion Matrix of NCC model train and tested with nRC dataset	51
29	Proposed model for future work	53
30	Proposed model for future work, consisting of multiple binary classifiers	54

List of Tables

1	Rfam MySQL Database - Connection Details	31
2	Number of Sequences per family per Data source	34
3	Number of Sequences per family in NCC dataset	35
4	nRC [4] dataset divided to Train and Test	37
5	One hot encoding of RNA Bases	41
6	One hot encoding of RNA IUPAC	41
7	NCC model training parameters	45
8	Test metrics using the NCC dataset with 4 and 8 digits RNA Base encoding	49
9	Models comparison.	50

1 Introduction

Non coding RNAs (ncRNAs) encompass an array of RNA molecules that aren't involved in protein encoding. Despite their lack of coding ability ncRNAs play roles, in regulating biological processes, including gene expression, DNA replication and epigenetic control. The identification of non coding RNAs (ncRNAs) has posed significant challenges for researchers studying their functions. Precisely predicting families is vital for advancing research on their functions due to the functions associated with different ncRNA families. Traditional experimental methods [5] used to identify families are not time consuming and labor intensive but also expensive making them impractical for high throughput technology demands. Consequently computational approaches have become indispensable in predicting families. Current methods, for predicting families can be broadly categorized into two groups:

- **Sequence or Secondary Structure-based Methods:** These methods rely on analyzing the sequence or secondary structure of ncRNAs to predict their families. While this approach can provide insights, the accuracy can be limited by the accuracy of predicted secondary structures.
- **Homologous Sequence Alignment Methods:** These methods [6] align ncRNAs to their homologs to identify shared features and predict their families. This approach often achieves high accuracy but requires consensus secondary structure annotation for ncRNAs and struggles to model complex structures like pseudoknots.

To address these challenges, NCC was developed, a new ncRNA classification tool that utilizes deep learning to analyze ncRNA sequences and classify them into their respective categories. The model processes RNA sequences directly, with no additional information required, converting them into a one-hot encoded format. It consists of four essential layers: a convolutional neural network layer for initial pattern recognition, followed by a pooling layer to reduce data size. Then, a bi-directional RNN layer examines the sequence from both ends for comprehensive analysis. The sequence concludes with a fully connected layer that synthesizes the findings into a final output, efficiently analyzing RNA sequences. The main goal of NCC is a simple model architecture for fast training purposes with high enough accuracy.

This thesis is designed to serve as a foundational resource for researchers, offering guidance and inspiration for the creation of future generations of non-coding RNA (ncRNA) classification tools. By presenting a comprehensive overview of current methodologies, challenges, and advancements in the field of ncRNA classification, this work aims to equip scientists and bioinformaticians with the knowledge and insights needed to innovate and develop more advanced, accurate, and efficient ncRNA classifiers. The ultimate goal is to propel the field forward, fostering the development of novel approaches that will enhance our understanding and analysis of ncRNA functions and their roles in complex biological processes.

1.1 Document Structure

The document, focused on the classification of non-coding RNAs (ncRNAs) using a novel deep learning model, is structured to guide the reader through a thorough understanding of RNA biology, computational approaches, and the practical application of artificial intelligence in genomic research. It begins with "RNA - Theoretical Background" section provides a deep dive into the biology of RNA, distinguishing between its various types and elaborating on the importance of non-coding sequences. In the "Artificial Intelligence" section, the document transitions to the computational methods, explaining the fundamentals of machine learning and neural networks as they apply to RNA classification. The "Technologies" section outlines the software and frameworks used, while "Related Work" places this research in the context of existing related studies. The "ncRNA Data" section details the datasets employed, followed by "Implementing the Prediction Mechanism," which describes the development and evaluation of the classification model. The document concludes with summaries of the findings, potential future research directions, and a comprehensive bibliography, offering a clear roadmap from theoretical underpinnings to practical applications in RNA research.

2 RNA - Theoretical Background

2.1 RNA

RNA, which is short, for ribonucleic acid is a molecule in biological processes. It plays a role in protein synthesis, gene expression and the regulation of information. Unlike DNA RNA is a stranded molecule composed of nucleotides. Each nucleotide consists of a base (adenine, cytosine, guanine or uracil) a sugar called ribose and a phosphate group. To put it simply RNA is synthesized from DNA through transcription. Helps translate the code into proteins. The cell relies on types of RNA such as messenger RNA (mRNA) transfer RNA (tRNA) ribosomal RNA (rRNA) and small nuclear RNA (snRNA) to carry out functions that are vital, for the organisms overall health and proper functioning.

- Messenger RNA (mRNA) is produced from DNA. Transports the information, from the nucleus to the ribosomes present in the cytoplasm. At the ribosomes it undergoes translation resulting in protein synthesis.
- Transfer RNA (tRNA) is a RNA molecule that carries acids to the ribosomes during protein synthesis. It accomplishes this task by matching acids with mRNA codons, which are three nucleotide sequences that encode particular amino acids. Subsequently it adds these acids to the growing chain.
- Ribosomal RNA (rRNA) forms a part of ribosomes which're cellular structures responsible for protein synthesis. Apart from providing support to ribosomes rRNA also plays a role in catalyzing chemical reactions involved in protein production.
- Small nuclear RNA (snRNA) belongs to a class of RNA molecules that contribute to cellular processes. One of its functions is participating in RNA splicing, which involves removing coding sequences called introns, from mRNA and joining together coding sequences known as exons.

Apart, from the RNA types mentioned earlier there are RNA molecules that have crucial regulatory functions, within cells. These include microRNA (miRNA) and small interfering RNA (siRNA). Their role primarily involves controlling gene expression by either inhibiting genes or breaking down mRNA molecules. RNA in general has a role to play when it comes to the movement of information and the control of gene expression, within cells. It is a molecule that's necessary, for the sustenance of life and is actively engaged in numerous significant biological processes.

2.2 Non-coding RNA

In the area of cellular mechanisms, a specific form of RNA known as non coding RNAs [7] (ncRNAs) have emerged as pivotal players. Traditionally RNA was perceived as a messenger, for conveying instructions, from DNA to create proteins. However, current understanding reveals that ncRNAs despite not being protein producing entities themselves, they play roles in regulating gene activity. Contribute significantly to diverse biological processes.

Non coding RNAs are a group that can be broadly classified based on their size and function. Small ncRNAs, which are usually than 200 nucleotides long include microRNAs (miRNAs) and small interfering RNAs (siRNAs) both of which're crucial for RNA interference—a critical process in silencing genes after transcription. Another subgroup called coding RNAs (lncRNAs) which exceed 200 nucleotides in length have diverse regulatory functions such as modifying chromatin structure and controlling transcriptional activities. Additionally, ribosomal RNAs (rRNAs) and transfer RNAs (tRNAs) well known for their involvement, in protein synthesis are also considered ncRNAs. This highlights the presence of coding sequences within the genome.

In Rfam [8] there are 13 classes of non-coding RNA families. In Figure 1 the leaves of the hierarchical tree represent ncRNA classes of Rfam, used in several previous researches as well as in this study. As stated in [9] 10% of RNACentral [10] sequences do not match any existing Rfam family and could potentially represent new families. Some of these sequences belong to non-coding RNA classes that are not suitable for inclusion in Rfam because they are unstructured, or their sequence length is too short or too long to be effectively modeled using CMs, such as piRNAs, rasiRNAs, siRNA, and some long non-coding RNAs (5% of the total number of sequences).

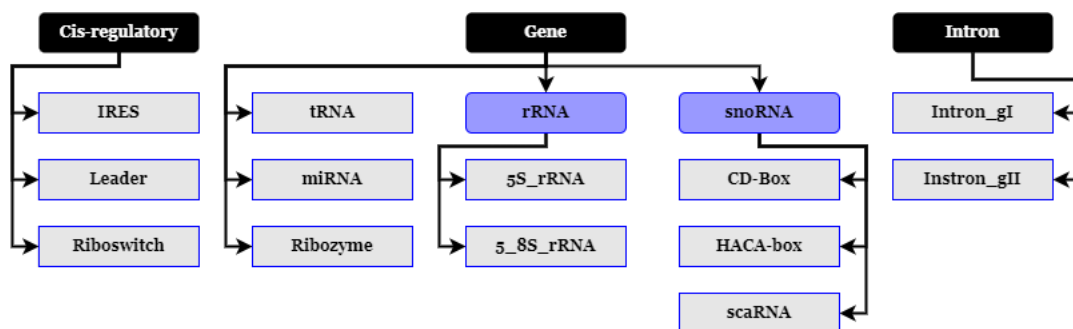


Figure 1: Non-coding RNA Classes and sub-Classes

The range of functions that coding RNAs (ncRNAs) perform is as diverse, as their classification. For example miRNAs play a role in controlling gene expression by binding to sequences on target mRNAs. This binding can lead to either degradation of the mRNA or inhibition of its translation into proteins. Such post transcriptional regulation is essential for processes like development, cell growth and programmed cell death (apoptosis). On the hand, lncRNAs have a range of functions including acting as molecular scaffolds that assist in the formation of ribonucleoprotein complexes or acting as decoys that divert proteins away from their intended target genes. The ability of lncRNAs to regulate gene expression at levels underscores their importance in maintaining balance.

Abnormal expression and malfunction of ncRNAs have been associated with diseases such as cancer, neurological disorders and cardiovascular diseases. For instance the dysregulation of miRNAs has been linked to tumor progression spread (metastasis) which makes them potential biomarkers for diagnosing and predicting cancer outcomes. Similarly, changes in patterns of expression have been observed in disorders like Alzheimer’s disease and schizophrenia, suggesting a potential role for

these molecules in the development of these conditions. The connection, between dysregulation and diseases, highlights the opportunities tied to targeting ncRNAs.

Once considered insignificant parts of our coding RNAs now take center stage in our understanding of how genes are regulated. Their various functions, in processes and the development of diseases present opportunities for further investigation and therapeutic interventions. As we delve deeper into the world of non-coding RNAs their potential to transform the field of medicine and advance our comprehension of biology is vast. This thesis underscores the importance of ncRNAs in the molecular orchestration of life, marking a paradigm shift in our perception of the genomic landscape.

2.3 RNA Secondary and Tertiary Structure

The secondary structure of an RNA [11] molecule refers to the arrangement of its nucleotides into base pairs and single strands, which determines its three-dimensional shape. RNA secondary structure can take several type of forms, including stem loop structures hairpin loops and pseudoknots. These formations occur when complementary nucleotides (adenine with uracil and cytosine with guanine) form base pairs and are stabilized by hydrogen bonds.

Understanding the secondary structure of RNA is crucial because it influences its functionality. For instance, transfer RNA (tRNA) relies on its structure to transport amino acids to the ribosome during protein synthesis. While the secondary structure of ribosomal RNA (rRNA) is important for its structural and catalytic role in the ribosome. Other types of RNA like microRNA (miRNA) and long non coding RNA (lncRNA) may also have important functions within cells that depend on their secondary structures.

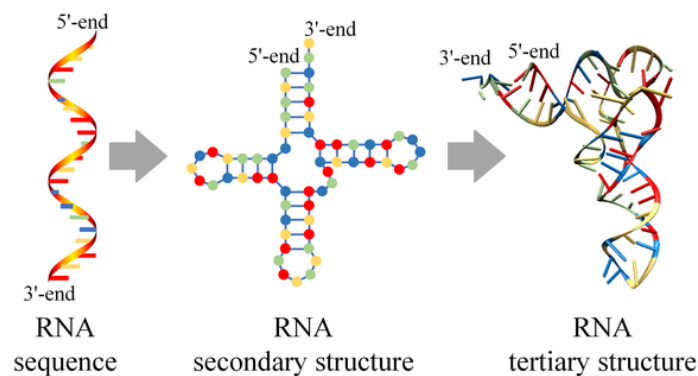


Figure 2: RNA Secondary and Tertiary structure (Image source [1])

Predicting the secondary structure of an RNA molecule is important for understanding its function and for designing RNA molecules with specific functions. There are several computational tools that can be used to predict the secondary structure of an RNA molecule, including RNAcon [12], IPknot [13] and Knotify [2]. These tools use various algorithms and models to predict the most likely secondary structure based on the nucleotide sequence of the molecule.

RNA tertiary structure denotes the three shape-dimensional shape that an RNA molecule folds, which is pivotal for its operation. Unlike the arrangement of nucleotides (primary structure) or simple loops and helices (secondary structure) the tertiary structure involves more complex folding, where distinct parts of the RNA molecule bend and loop back onto themselves. This folding process is influenced by interactions between components of the RNA molecule including bonds between nucleotides that are distantly positioned in the sequence. The resulting shape, from this folding determines how the RNA functions, whether it participates in protein synthesis, gene expression regulation or other cellular mechanisms.

3 Artificial Intelligence

Artificial Intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to "think" and "learn" like humans. Essentially, it's about creating smart machines capable of performing tasks that typically require human intelligence. This can include things like understanding natural language, recognizing images, solving problems, and making decisions. AI is integrated into various aspects of our daily lives, from the virtual assistants in our smartphones to more complex systems like autonomous vehicles. Its aim is not just to automate tasks, but also to make machines capable of adapting to new situations, a trait that's inherently human. This technology is continually evolving, making our interaction with digital systems more intuitive and efficient.

3.1 Machine Learning

Machine Learning is a form of Artificial Intelligence that enables computers to acquire knowledge from experiences, like how humans learn from their own actions. It involves training computers using volumes of data and algorithms that enable them to grasp how to carry out a task. For instance, machine learning can be utilized for speech recognition, movie recommendations and even weather prediction. What distinguishes this approach is that these machines are not explicitly programmed for these tasks; instead, they enhance their abilities over time by analyzing data. This makes machine learning a tool, for tackling problems and enhancing the intelligence and user-friendliness of our technology.

- **Supervised learning** stands as a fundamental approach where the algorithm is trained on a pre-labeled dataset, akin to a student learning under the guidance of a teacher. This method involves providing the algorithm with both the input (such as images) and the desired output (the corresponding labels identifying the content of the images). The objective is to enable the algorithm to discern patterns and relationships between the input and output. Through this training process, the machine becomes equipped to analyze new, unlabeled data and predict or classify it accurately. Supervised learning is integral to numerous applications, including email spam detection, where the system is trained to differentiate between spam and non-spam emails, thereby optimizing the filtering process.
- **Unsupervised learning**, another crucial aspect of machine learning, involves training algorithms using data that is not labeled or categorized. Unlike supervised learning, where the model is guided by known outcomes, unsupervised learning allows the algorithm to analyze and organize data autonomously. The machine identifies patterns, correlations, and anomalies within the data, without any prior instruction on what to look for. This method is particularly useful for discovering hidden structures in data, such as grouping customers with similar buying habits in marketing analysis or detecting unusual patterns that could indicate fraudulent activity. Unsupervised learning paves the way for machines to uncover insights from datasets that are too complex or vast for human analysis.

- **Semi-Supervised Learning** is a technique, in machine learning, that combines elements of unsupervised and supervised learning. This method involves training the algorithm using a set of labeled data along with a pool of unlabeled data. By blending the strengths of both types of learning, it aims to achieve precision while minimizing the need for labeled data and benefiting from the exploratory capabilities of unsupervised learning. This approach is particularly valuable when acquiring labeled data is expensive or time consuming. Leveraging both labeled and unlabeled data allows for accuracy and efficiency in the learning process. Semi supervised learning finds applications in domains such, as image and speech recognition where labeling datasets can be impractical but having a small amount of labeled data can greatly assist and enhance the learning process.

3.2 Classification Algorithms

In the field of machine learning, classification problems revolve around algorithms that sort and categorize data into established classes or groups. The main objective is to assign unfamiliar data to one of these predetermined categories based on its distinctive characteristics. An example we often encounter is email filtering, where incoming emails are classified as either "spam" or "non spam". In this scenario the algorithm learns from a training dataset that already contains labeled emails, as either spam or non spam. It then utilizes this knowledge to classify emails. Classification problems can be binary involving two classes like spam detection. Multi class involving than two classes like identifying various types of fruits in images. This approach plays a role in a range of applications, including medical diagnoses (classifying diseases) and financial analysis (identifying different credit risk categories). The effectiveness of a classification model is typically assessed by evaluating its accuracy in predicting the category, for acquired data.

3.3 Artificial Neural Networks

Artificial Neural Networks (ANNs) belong to a category of machine learning techniques that draw inspiration from the structure and function of the brain. These networks consist of interconnected nodes, often referred to as "neurons", which perform calculations. Data enters through an input layer, passes through one or more layers where processing occurs and eventually exits through an output layer. The weights connecting these neurons representing the strength of their connections are adjusted during the learning process.

ANNs excel at handling problems that involve linear relationships. They learn by tuning these weights based on how much they deviate from predictions in a process known as backpropagation. This adaptive mechanism allows them to make predictions or decisions as they are exposed to more data.

These networks possess versatility. Find applications, in various domains such as image and speech recognition natural language processing and even complex game playing. ANNs ability to learn from datasets and uncover patterns that may not be immediately apparent to humans makes them a formidable tool in the field of AI.

Their architecture and functionality continue to evolve, enabling them to tackle an expanding range of tasks efficiently.

3.3.1 Neuron

A neuron in a neural network, often referred to as a node, a unit or a cell, is a fundamental building block that mimics the function of neurons in the human brain, albeit in a simplified manner. Each neuron in a neural network functions as a tiny processing unit, performing simple calculations and contributing to the network's ability to solve complex problems.

Key aspects of a neuron in a neural network include:

- **Inputs:** Each neuron receives input from either the original dataset or the output of other neurons. In the context of deep learning, these inputs can be features of the data, such as pixels in an image or words in a text.
- **Weights:** Every input is associated with a weight, which is a trainable parameter of the model. These weights determine the influence of the input on the neuron's output. During the training process, these weights are adjusted to help the model make accurate predictions or decisions.
- **Activation Function:** Once the inputs have been multiplied by their respective weights and summed, the total is passed through an activation function. This function determines whether and to what extent the signal should progress further through the network. Common activation functions include sigmoid, tanh, and ReLU (Rectified Linear Unit).
- **Output:** The result of the activation function is the output of the neuron. This output can either be used as an input to neurons in the next layer of the network or, in the case of the final layer, as the final output of the neural network for a given input.
- **Bias:** A bias term is often added to the input sum before applying the activation function. This term allows the neuron to shift the activation function to the left or right, which can be critical for successful learning.

Perceptron A Perceptron is a type of neuron and one of the basic versions of a neural network. It was initially created by Frank Rosenblatt in the 1950s. Essentially it acts as a component, for intricate types of neural networks and provides a fundamental framework, for comprehending how neural networks operate.

$$y = f \left(\sum_{i=1}^n w_i x_i + b \right) \quad (1)$$

- **y :** This is the output of the Perceptron. It's the final result after all the calculations are done, representing the classification made by the Perceptron. For a simple binary classifier, y will typically be 1 or 0, indicating which of the two possible classes the input has been assigned to.

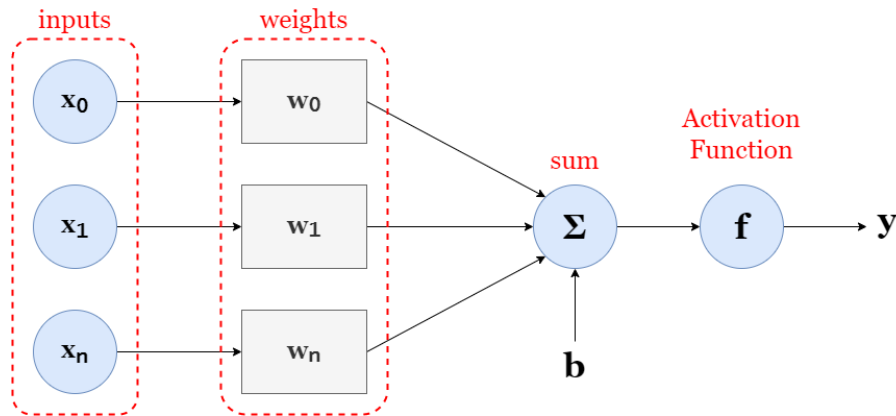


Figure 3: Diagram of perceptron neuron

- f : This stands for the activation function. In the context of a Perceptron, this is often a step function. The activation function takes the weighted sum of the inputs plus the bias and converts it into an output. For a step function, if the sum is above a certain threshold, the output is 1; if it's below the threshold, the output is 0.
- w_i : These are the weights. Each input feature x_i has a corresponding weight w_i associated with it. These weights are parameters that the Perceptron learns during the training process. They determine how much influence each input will have on the final output. The learning process of a Perceptron involves adjusting these weights based on the error in its predictions.
- x_i : These are the input features. In the context of a dataset, each x_i could be a different attribute or characteristic of the data. For example, in a dataset of houses, x_i could represent features like size, number of rooms, or age of the house.
- b : This is the bias term. The bias allows you to shift the activation function to the left or right, which can be critical for successful learning. It's like an extra input to the Perceptron that always has the value 1 but has its own weight. The bias ensures that even when all the input features are zero, the neuron can still produce a non-zero output.
- n : This represents the number of inputs to the Perceptron. If you have a dataset with many features, n will be large, indicating that the Perceptron is taking into account many different factors to make its classification.

3.4 Activation Functions

Activation functions [14] in neural networks are crucial as they introduce non-linear properties to the network, enabling it to learn and perform more complex tasks that a linear equation couldn't. Without these functions, the neural network would essentially become a linear regression model, incapable of handling the complexities found in real-world data.

1. **Binary Step Function** This function is a threshold-based activation function. It outputs a binary result based on whether the input value is above or below a certain threshold.

$$f(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

2. **Linear Activation Function** This function is simply a linear equation of the input. It returns the input as is, without applying any threshold or transformation

$$f(x) = ax \quad (3)$$

3. **Sigmoid or Logistic Function** It outputs values between 0 and 1, making it useful for binary classification. However, it suffers from the vanishing gradient problem, where neurons with inputs of large magnitude saturate at 0 or 1, with almost no gradient for backpropagation.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (4)$$

4. **Hyperbolic Tangent (tanh)** Similar to the sigmoid but outputs values between -1 and 1. It's zero-centered, making it easier for the model to converge and learn, but it still suffers from the vanishing gradient problem for very high or very low values of x.

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (5)$$

5. **Rectified Linear Unit (ReLU)** It outputs x if x is positive; otherwise, it outputs zero. ReLU is very popular because it overcomes the vanishing gradient problem, allowing models to learn faster and perform better. However, it can suffer from the "dying ReLU" problem, where neurons can become inactive and stop contributing to the model learning.

$$\text{ReLU}(x) = \max(0, x) \quad (6)$$

6. **Leaky ReLU** A variation of ReLU, it allows a small, non-zero gradient when the unit is not active, preventing neurons from dying.

$$\text{ReLU}(x) = \max(0.01x, x) \quad (7)$$

7. **Softmax Function** Typically used in the output layer of a classifier, it turns logits (raw predictions) into probabilities that sum up to 1. This function is particularly useful for multi-class classification problems.

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \quad (8)$$

3.5 Deep learning

Deep learning is an advanced subset of machine learning that utilizes the use of complex neural networks. These networks consist of many layers, hence the term "deep", and they process data in a highly sophisticated manner. The design of learning models enables them to comprehend and interpret quantities of data resembling the cognitive processes of humans, in a more direct and focused manner.

3.6 Fully Connected layer

A Fully Connected (FC) layer is a key component in neural networks, particularly in Deep Learning architectures. It's called "fully connected" because every neuron in this layer is connected to every neuron in the previous layer. This dense network of connections allows the FC layer to integrate and process information learned by the network in earlier layers.

In practical terms, the FC layer's role is to take the outputs from the previous layers, which often represent learned features from the input data (like edges or textures in an image), and combine them to form the final output. For instance, in a facial recognition task, while the earlier layers might identify features like edges, colors, and textures, the FC layer combines these to recognize specific faces.

The FC layer is usually placed towards the end of the neural network and is often followed by a classification layer, like a softmax, which is used to predict probabilities of different classes. The strength of a fully connected layer is in its ability to learn non-linear combinations of the high-level features extracted by previous layers, making it integral in transforming these features into complex outputs like classifications or predictions.

3.7 Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN) is a kind of network that is mainly used for processing data that has a grid like structure, like images. CNNs are renowned for their effectiveness in image recognition and processing tasks.

The main element of a CNN is the convolutional layer. This layer applies a series of filters to the input data to create feature maps. These filters systematically scan the input and perform a mathematical operation called a convolution, which essentially means multiplying and summing up values. This process helps the network focus on specific features like edges, textures, or shapes. After the convolutional layers, there are pooling layers, which reduce the spatial size of the feature maps. This downsampling helps reduce the amount of computation and parameters in the network, making it more efficient.

CNNs also typically include fully connected layers towards the end, which integrate the learned features from the convolutional and pooling layers to perform classification tasks, like identifying objects in images.

CNNs are powerful when it comes to tasks that involve the recognition and classification of visual data, being able to capture and learn patterns from images with high efficiency and accuracy. They are widely used in applications like image

and video recognition, image classification, medical image analysis, and even in self-driving cars for visual perception.

3.7.1 Convolution Layer

The convolutional layer plays a role in the architecture of a Convolutional Neural Network (CNN). Its main task is to capture characteristics from input data, images. This layer employs filters also referred to as kernels to examine the input data and carry out convolution operations.

1. **Filters/Kernels:** A filter is a small matrix of weights. Each filter is designed to detect specific features, such as edges, textures, or patterns in the input data.
2. **Convolution Operation:** The filter is applied to the input data by sliding it across the input image. At each position, a calculation is made by multiplying the filter and the corresponding section of the image. This process combines the filter values, with the values of the image to effectively extract features from the area of the image.
3. **Feature Maps:** The result of applying a filter across the entire input image is a feature map. This map represents the locations and strengths of detected features in the input. For instance, if the filter is designed to detect vertical edges, the feature map will have high values in regions with vertical edges.
4. **Stride and Padding:** The stride determines the movement of the filter across the image. When the stride is set to 1 the filter moves pixel by pixel, while a larger stride moves it faster, skipping pixels. Padding refers to adding pixels (zeros) around the input image to ensure that the filter can be effectively applied to all areas of the image including its borders.
5. **Non-Linearity (Activation Function):** After the convolution operation, an activation function like ReLU (Rectified Linear Unit) is typically applied to introduce non-linearity, allowing the network to learn more complex patterns.
6. **Multiple Filters:** A convolutional layer usually contains multiple filters, each detecting different features. When applied to the input, this creates a stack of feature maps, one for each filter, providing a comprehensive representation of various features in the input.

3.7.2 Pooling Layer

The pooling layer plays a role, in Convolutional Neural Networks (CNNs). Usually comes after one or more convolutional layers. Its main job is to decrease the dimensions (width and height) of the input volume for the convolutional layer. This reduction in dimensionality serves several purposes:

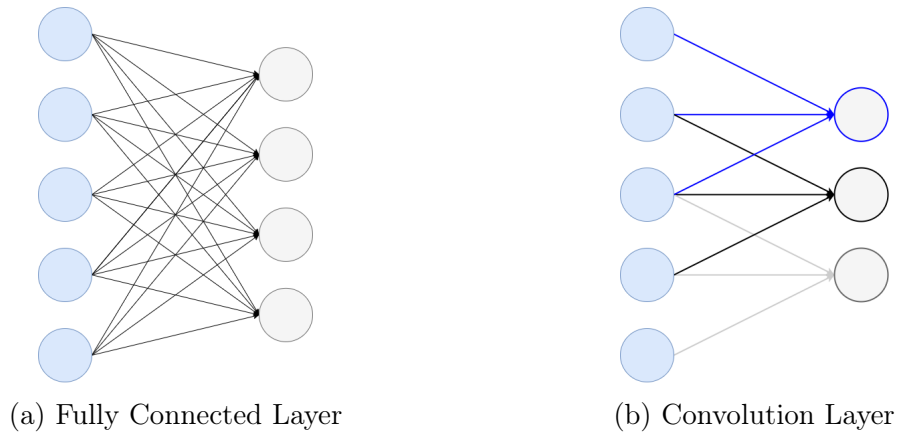


Figure 4: Fully connected (a) and convolution (b) layer example

1. **Reduced Computation:** By downsizing the input data, pooling layers decrease the number of parameters and computations in the network, leading to improved computational efficiency.
2. **Avoiding Overfitting:** Pooling helps in preventing overfitting, a common problem in machine learning where a model performs well on training data but poorly on unseen data. By reducing the complexity of the model, it helps in generalizing the patterns learned.
3. **Maintaining Spatial Invariance:** Pooling helps the network to become invariant to slight changes and distortions in the input image, making it better at recognizing objects irrespective of their variations in position and orientation in the image.

The most common types of pooling are:

- **Max Pooling:** This method involves selecting the maximum element from the region of the feature map covered by the filter. For example, in a 2x2 pooling filter, max pooling would take the largest element from the 2x2 region. Max pooling is effective in capturing the most prominent feature in the local patch of the feature map.
- **Average Pooling:** Instead of taking the maximum value, average pooling computes the average of the elements in the region of the feature map covered by the filter.

The pooling layer typically operates independently on each depth slice of the input and resizes it spatially. The use of pooling layers in CNN architectures is a critical factor in their ability to efficiently process high-resolution images and video.

3.8 Recurrent Neural Network (RNN)

A recurrent neural network (RNN) is a kind of neural network (ANN) that is specifically designed to handle data that occurs in a sequence. Unlike feedforward net-

works, which process data in a linear manner RNNs have loops that enable information to flow back through the network. This unique feature allows RNNs to effectively model and understand relationships and patterns, in data. The way RNNs operate is by analyzing the input data while considering the information from time steps through the recurrent connections. This mechanism empowers the network to learn and capture patterns in data, such as the word order, in a sentence or the melodic structure of a musical composition.

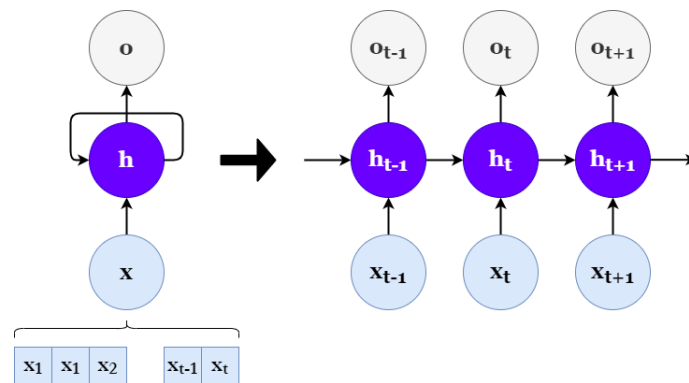


Figure 5: RNN node Unfolding

RNNs have a wide range of applications, particularly in tasks that involve sequential data. Here are some of their notable applications:

- **Natural Language Processing (NLP):** RNNs are widely used in NLP tasks, such as machine translation, text summarization, sentiment analysis, and chatbot development.
- **Speech Recognition:** RNNs are employed in speech recognition systems to convert spoken language into text. By analyzing the temporal patterns of speech, RNNs can identify words and phrases.
- **Music Generation:** RNNs can generate new music by learning patterns from existing compositions. They can also create variations on existing melodies or improvise new ones.
- **Time Series Forecasting:** RNNs can be used to forecast future values of a time series, such as stock prices, weather patterns, or customer behavior. They analyze historical data to identify trends and patterns that can inform predictions.
- **Drug Discovery:** RNNs are being explored in drug discovery to analyze molecular structures and predict interactions between drugs and proteins. This could lead to the development of more effective and targeted therapies.

One of the difficulties encountered with RNNs is the issue of maintaining long term connections. RNNs often face challenges when it comes to preserving information, over extended time intervals during training resulting in problems such as vanishing gradients. To tackle this problem advanced forms of RNNs, LSTMs (Long Short Term Memory networks) and GRUs (Gated Recurrent Units) have been developed. These models are specifically designed to retain long term dependencies.

3.8.1 Long-Short term memory (LSTM)

LSTM networks [15], known as Long Short Term Memory networks are a kind of Recurrent Neural Network (RNN) that tackle the problem of learning long term dependencies. Unlike RNNs, which excel at capturing short term dependencies but face difficulties in retaining information over long sequences and encounter issues like vanishing or exploding gradients LSTMs overcome these challenges using a distinctive architecture that enables better retention of information, over extended periods.

The main concept, behind an LSTM is its cell state. This state is responsible for storing and transferring information across the network facilitating the management of long term dependencies. LSTMs are equipped with three types of gates (Figure 6) that control the movement of information into and, out of the cell state:

- The **forget gate** f_t decides what information should be discarded from the cell state.
- The **input gate** i_t determines what new information should be added to the cell state.
- The **output gate** o_t decides what part of the current cell state should be used to generate the output at the current time step.

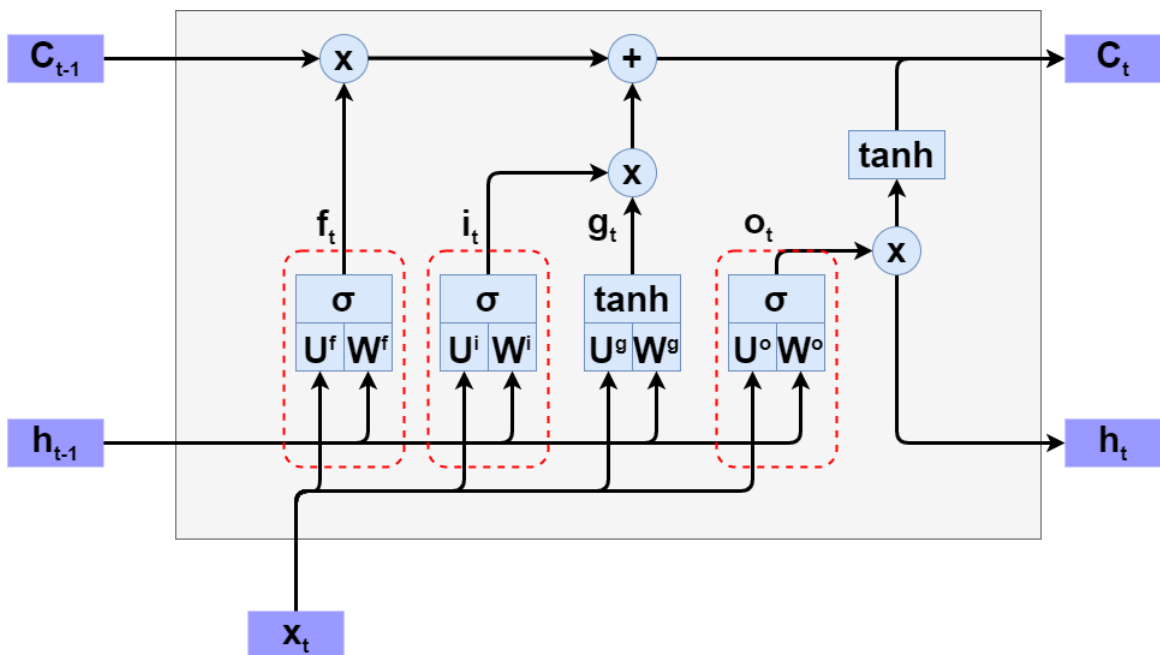


Figure 6: LSTM cell Architecture

These gates allow the LSTM to selectively remember or forget things, making it very efficient at learning from long sequences of data. This is particularly useful in applications like language modeling, where understanding the context from a long passage of text is crucial.

Where:

$$\begin{aligned}
\sigma(x) &= \frac{1}{1 + e^{-x}} \\
\tanh(x) &= \frac{e^x - e^{-x}}{e^x + e^{-x}} \\
i_t &= \sigma(x_t U^i + h_{t-1} W^i) \\
f_t &= \sigma(x_t U^f + h_{t-1} W^f) \\
o_t &= \sigma(x_t U^o + h_{t-1} W^o) \\
g_t &= \tanh(x_t U^g + h_{t-1} W^g) \\
C_t &= \sigma(f_t C_{t-1} + i_t g_t) \\
h_t &= \tanh(C_t) o_t
\end{aligned} \tag{9}$$

3.8.2 Gated Recurrent Unit (GRU)

A Gated Recurrent Unit (GRU), introduced in [16], is a type of Recurrent Neural Network (RNN) that is designed to capture patterns in sequential data, similar to Long Short Term Memory (LSTM) networks. However GRUs have a simpler structure (Figure 7) compared to LSTMs. They are known for their efficiency in modeling sequences and their ability to overcome the vanishing gradient problem commonly encountered in RNNs. Unlike LSTMs, which have three gates (input, output and forget) GRUs only have two gates, the update gate and the reset gate. This simplification results in fewer parameters and a more streamlined model enabling faster training without significantly sacrificing performance.

1. **Update Gate:** The update gate z_t in a GRU controls the extent to which a new state overwrites the old state. It's a combination of the forget and input gates in an LSTM, deciding how much of the previous (past) information needs to be passed along to the next (future).
2. **Reset Gate:** The reset gate r_t is used to decide how much of the past information to forget. It allows the model to drop any irrelevant information in the future (next) steps, effectively resetting the memory of the network.

Where:

$$\begin{aligned}
r_t &= \sigma(x_t U^r + h_{t-1} W^r) \\
z_t &= \sigma(x_t U^z + h_{t-1} W^z) \\
g_t &= \tanh(x_t U^g + r_t W^g h_{t-1}) \\
h_t &= (1 - z_t) g_t + z_t h_{t-1}
\end{aligned} \tag{10}$$

3.8.3 Bidirectional RRN

A Bidirectional Recurrent Neural Network (BiRNN) [17] expands upon the Recurrent Neural Network (RNN) by incorporating a mechanism to process data, in both forward and backward directions (Figure 8). This approach equips the network with access to both information about the sequence at each point. In a BiRNN there exist

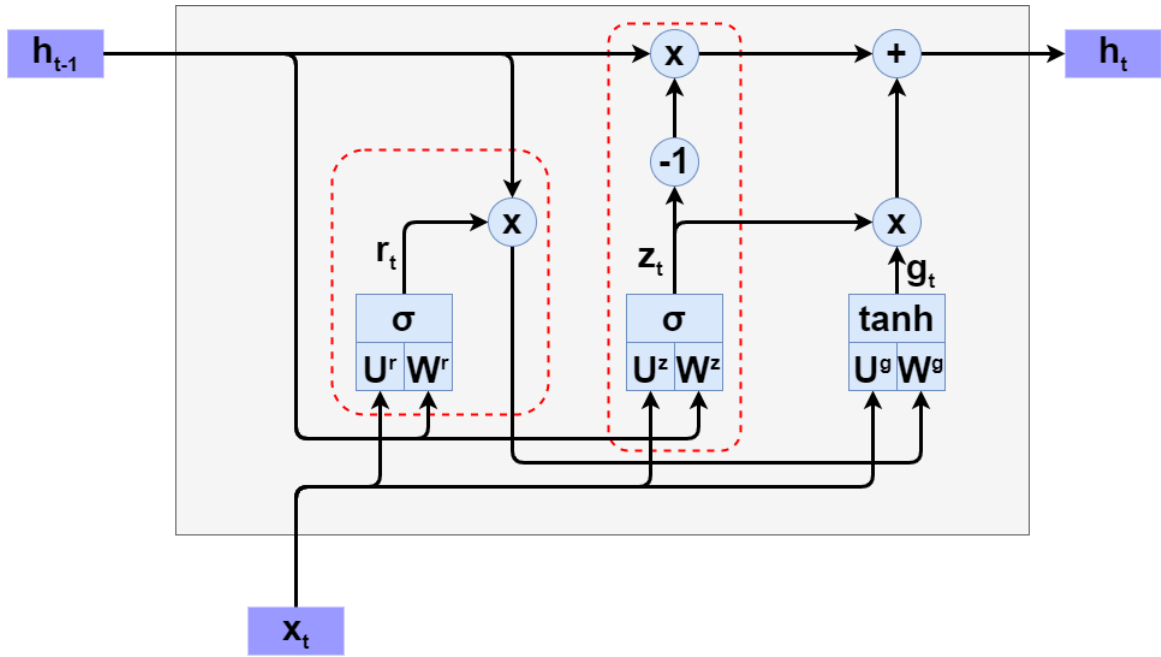


Figure 7: GRU cell Architecture

two layers of RNNs. One layer handles the data in a direction from the beginning to the end of the sequence, while the other layer handles it in a direction from the end to the beginning. This dual layer structure empowers the network to capture insights, from both forthcoming states.

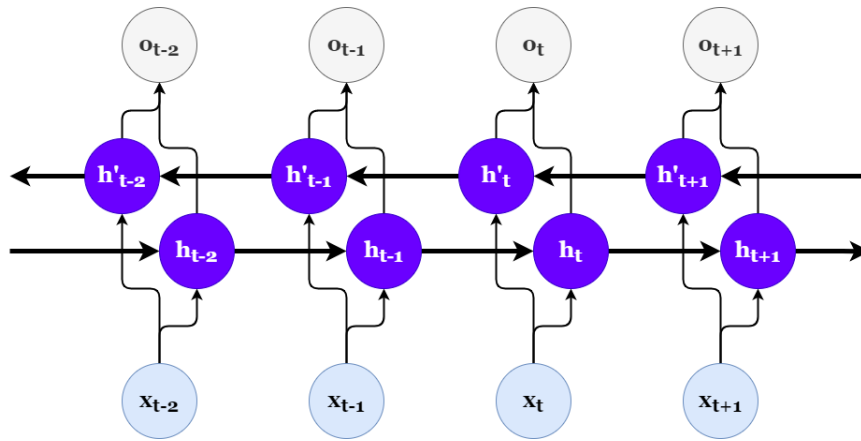


Figure 8: Layer of a Bidirectional RNN

4 Technologies

4.1 Python

Python is widely regarded as a choice, for developing AI models due to it's because it's easy to learn and use. Its straightforward syntax simplifies coding AI algorithms. Python also offers support for specialized tools and libraries specifically designed for AI, such as TensorFlow, Keras and PyTorch. These resources greatly facilitate the process of building AI models. Moreover, Python boasts a community of users who actively collaborate, exchange ideas and provide assistance to one another. This collaborative environment makes it easier to find solutions and discover techniques in the realm of AI. Given these factors it's no surprise that Python has gained popularity and proves valuable for AI projects.

4.2 Deep learning Frameworks

4.2.1 Tensorflow

TensorFlow [18], developed by Google is an open source framework, for machine learning. It is known for its computational graph paradigm, where computations are represented as a directed graph. Originally, TensorFlow used a static computation graph, meaning that the entire structure of the graph had to be defined before any computation could take place. However, TensorFlow 2.x introduced eager execution, allowing for flexibility and intuitive development. It supports hardware options such as CPUs, GPUs and TPUs (Tensor Processing Units). With an active community, TensorFlow offers a wealth of resources and libraries, for machine learning tasks.

4.2.2 Keras

Keras [19] started as an independent high-level neural networks API, designed to be user-friendly and provide a simple interface for building and training models. Keras aims to provide an unfair advantage to any developer looking to ship Machine Learning-powered apps. Keras focuses on debugging speed, code elegance and conciseness, maintainability, and deployability. By using keras codebase is smaller, more readable, easier to iterate on. Keras models run faster thanks to XLA compilation with JAX and TensorFlow, and are easier to deploy across every surface (server, mobile, browser, embedded) thanks to the serving components from the TensorFlow and PyTorch ecosystems

4.2.3 PyTorch

PyTorch [20] is an open-source machine learning framework developed by Facebook's AI Research lab (FAIR). It is known for its dynamic computation graph, which allows for more flexible and intuitive development. Unlike TensorFlow's earlier versions, PyTorch embraces a more Pythonic and imperative style, making it popular among researchers and practitioners who prefer a dynamic approach to model construction. PyTorch gained significant popularity in the research community due to its ease of use, strong support for GPU acceleration, and a highly active community.

4.2.4 Key Differences

Computational Graphs

TensorFlow originally used static computation graphs, while PyTorch employs dynamic computation graphs. Keras, when integrated with TensorFlow, follows TensorFlow's computational graph paradigm.

Ease of Use

TensorFlow had a steeper learning curve initially, but with TensorFlow 2.x and eager execution, it became more intuitive. Keras has always been designed for ease of use and user-friendliness. PyTorch is known for its intuitive and Pythonic programming style.

Flexibility and Control

TensorFlow provides both high-level and low-level APIs, offering a balance between abstraction and control. Keras abstracts many low-level details, providing less flexibility compared to TensorFlow's lower-level APIs. PyTorch offers a highly flexible and dynamic approach, allowing for a high degree of control over models.

Community and Ecosystem

TensorFlow has a large and established community with extensive resources and third-party libraries. Keras benefits from TensorFlow's ecosystem and has its own community as well. PyTorch gained rapid popularity in research communities, particularly for its dynamic computation graph.

Hardware Support

TensorFlow provides extensive support for various hardware, including CPUs, GPUs, and TPUs. PyTorch is well-suited for GPU acceleration and is adaptable to different hardware setups.

Ultimately, the choice between TensorFlow, Keras, and PyTorch depends on individual preferences, project requirements, and familiarity with the framework. All three are powerful tools with active communities, and they are widely used in both research and industry.

4.3 Other Python Libraries

4.3.1 NumPy

NumPy [21], short for Numerical Python, is a fundamental package for scientific computing in Python. It is renowned for its powerful N-dimensional array object, which is a versatile container for large and multidimensional arrays. NumPy arrays facilitate advanced mathematical and statistical operations, as they are optimized for performance and allow for efficient array processing. The library also provides tools for integrating C/C++ and FORTRAN code, enabling further optimization and speed. Its widespread popularity among data scientists and researchers stems from its high-level mathematical functions, ease of integration with other libraries,

and its pivotal role in the broader ecosystem of data analysis, machine learning, and scientific computing in Python.

4.3.2 Pandas

Pandas [22] is a Python library that is widely used for manipulating and analyzing data. It offers high-level data structures, like DataFrame and Series which simplify the handling and processing of information. With Pandas individuals can efficiently carry out tasks such as cleaning and transforming data merging datasets and creating visualizations. Its ability to import and export data in formats such as CSV, Excel and SQL databases has made it a preferred tool, among data scientists and analysts. The user-friendly interface of Pandas, coupled with its range of features, has firmly established it as a component of the Python data analysis toolkit.

4.3.3 Scikit-learn

Scikit learn [23], also known as sklearn is a used machine learning library, for Python that is highly regarded for its simplicity and efficiency. It offers an array of tools for machine learning tasks such as classification, regression, clustering and dimensionality reduction. Scikit learn is built upon the foundations of NumPy and SciPy while providing an interface to create and fine tune machine learning models. Its comprehensive documentation, user nature and ability to handle types of data make it a favored choice among both beginners and experienced practitioners, in the realm of data science and machine learning.

4.3.4 Matplotlib

Matplotlib [24] is a known Python library that is widely used for creating animated and interactive visualizations. It has gained popularity, for its user nature and flexibility allowing users to generate several types of plots and graphs using just a few lines of code. One of the advantages of Matplotlib is its level of customization, which grants users the ability to fine tune nearly every aspect of their plots. From axes limits and labels to the overall style and color scheme. This adaptability makes it an indispensable tool in Python for visualizing data in professional contexts serving purposes such, as exploring data patterns presenting research findings and depicting complex datasets.

4.4 RNA Secondary Structure Predict

4.4.1 Knotty

Knotty [25] is an algorithm for the efficient and accurate prediction of complex RNA pseudoknot structures. It improves on previous methods by handling a comprehensive class of pseudoknots with significantly reduced space complexity and enhanced prediction accuracy, leveraging a novel technique called sparsification. Knotty's performance, evaluated against other leading methods, showcases its superior capability in predicting biologically relevant pseudoknots with lower space and time

requirements. This advancement enables more practical applications in computational pseudoknot structure prediction, overcoming limitations of prior tools and offering a valuable case study on complex space-efficient algorithm optimization for RNA research.

4.4.2 IPknot

IPknot [13] is an innovative approach for predicting RNA secondary structures, particularly those involving pseudoknots, by focusing on optimizing the accuracy of the predicted structure. IPknot effectively breaks down a pseudoknotted structure into several pseudoknot-free components, while also estimating a base-pairing probability distribution that accounts for pseudoknots. This process enables IPknot to model various types of pseudoknots efficiently and operate at high speeds. Additionally, a heuristic algorithm was developed that refines base-pairing probabilities, enhancing IPknot's predictive precision. The optimization of expected accuracy is achieved through the use of integer programming combined with a threshold cut. Furthermore, IPknot has been adapted to predict a consensus secondary structure, inclusive of pseudoknots, from multiple sequence alignments.

4.4.3 Knotify

Knotify [2] is an Efficient Parallel Platform for RNA Pseudoknot Prediction Using Syntactic Pattern Recognition" presents a novel methodology for predicting RNA secondary structures, particularly focusing on RNA pseudoknots. The approach combines syntactic pattern recognition and context-free grammar (CFG) to predict RNA pseudoknots. High level representation of Knotify architecture is presented in Figure 9. The process involves parsing RNA sequences using CFG, identifying potential pseudoknot patterns, and employing a novel heuristic based on free-energy minimization to resolve ambiguities in the folding patterns. The method was tested using a dataset of 262 RNA sequences, achieving a recall ratio of 76.4% in predicting core stems of RNA pseudoknots. The F1-score and Matthew's Correlation Coefficient (MCC) were 0.774 and 0.543, respectively, outperforming other platforms in terms of execution time.

At this point is worth mentioning next version of Knotify

1. Knotify+ [26] framework introduces an innovative approach to predict H-type pseudoknots, which includes complex features like bulges and internal loops, by leveraging context-free grammar (CFG). By combining CFG's strengths with strategies for optimal base pairing and minimizing free energy, Knotify+ achieves significant performance improvements in predicting the core stems of pseudoknots. It not only surpasses existing frameworks in accuracy, especially for shorter sequences, but also offers competitive accuracy for longer sequences with reduced execution times, making it a cutting-edge solution in the field.
2. The updated framework [27] introduces a pruning technique that efficiently narrows down the grammar's search space. By filtering out trees that emerge from rare conditions, it achieves a significant reduction in execution time—33%

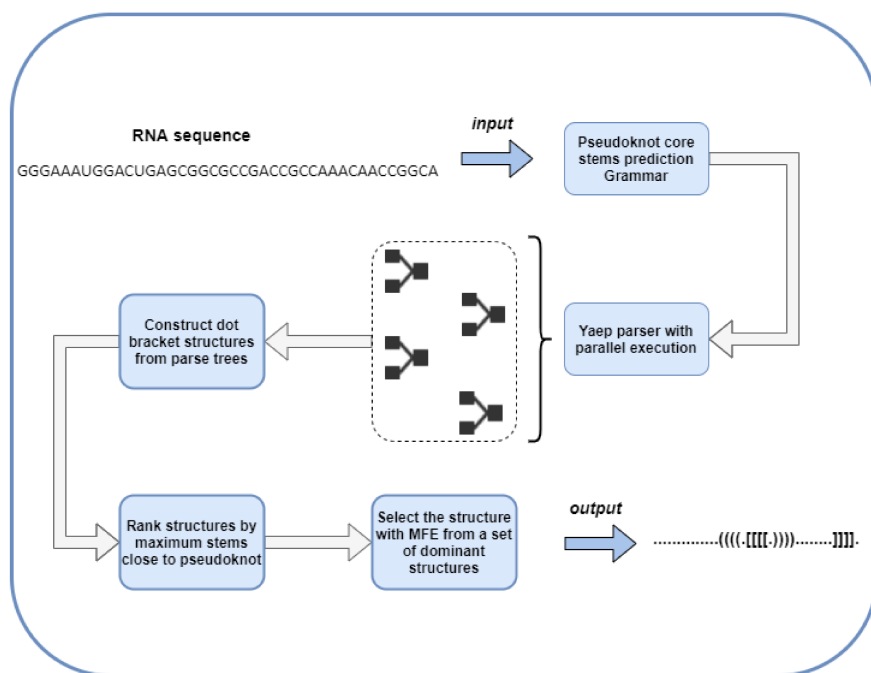


Figure 9: Knotify approach representation (Image Source [2])

faster than the original grammar-based method and 43% quicker than the brute-force approach—while maintaining the original level of accuracy.

3. This approach [28] introduces a grammar-based framework for predicting all L-type pseudoknots in a sequence efficiently, incorporating key biological concepts like maximum base pairing and minimizing free energy. Evaluating its effectiveness using four performance metrics: precision, recall, Matthews correlation coefficient (MCC), and the F1-score. This method outperformed three well-known methods in precision (0.844) and showed superior F1-score (0.671) and MCC (0.521), demonstrating its high accuracy and reliability. Added to RNA toolset, this methodology enhances biologists' ability to predict RNA motifs, offering potential applications in gene therapy, drug design, and understanding RNA functionality. It can also be combined with other methods to improve RNA structure prediction accuracy.

5 Related Work

5.1 RNAcon

RNAcon [12] is a tool that includes two different prediction models: one for distinguishing non-coding and coding RNAs, and another for classifying predicted non-coding RNAs into various categories. The model for distinguishing between non-coding and coding RNAs uses a machine learning approach based on Support Vector Machines (SVMs) and nucleotide composition as input features. To optimize and evaluate the model, three different datasets were used and various kernels and parameters of the SVM were tested using a 10-fold cross-validation technique. The final model implemented in the RNAcon web server uses tri-nucleotide compositions (TNC) for discrimination between non-coding and coding RNA sequences. RNAcon algorithm is presented in figure 10

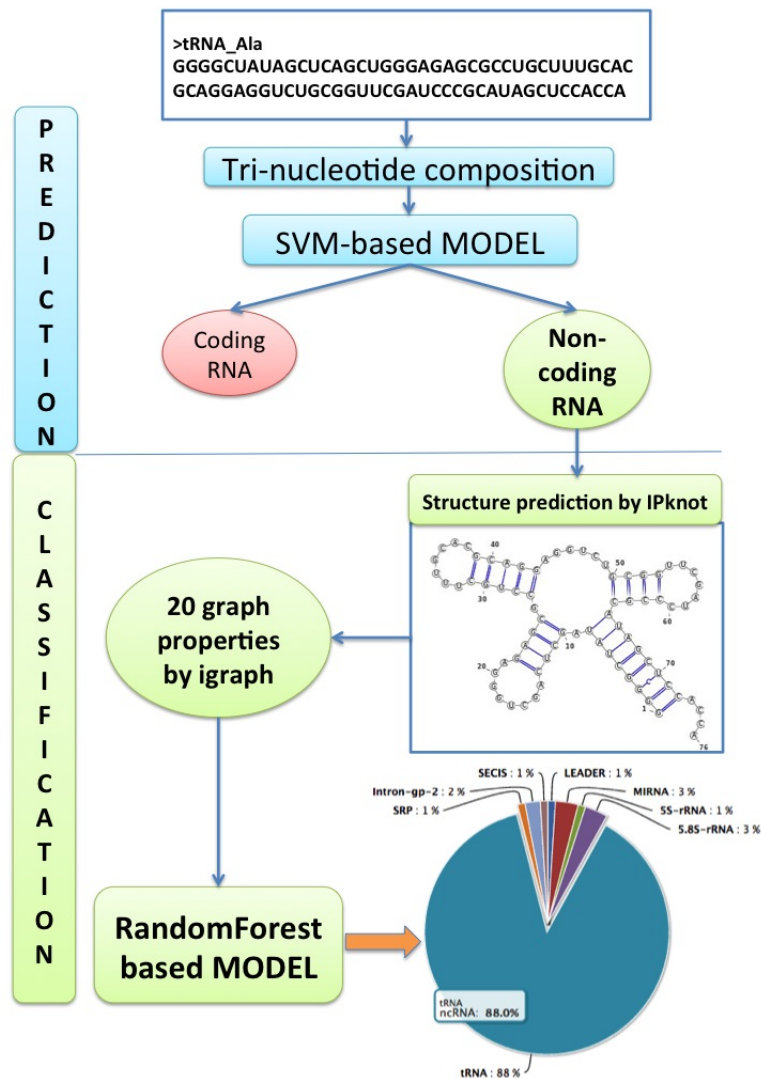


Figure 10: RNAcon Algorithm - a non-coding RNA Classifier [Image Source]

5.2 nRC

The nRC [4] tool is based on the extraction of features from the ncRNA secondary structure and a supervised classification algorithm using a deep learning architecture based on convolutional neural networks. The nRC tool was tested for the classification of 13 different ncRNA classes, similar to previous described RNAcon tool, and achieved an accuracy and sensitivity score of about 74%. The nRC tool outperformed other similar classification methods that have been developed until the year 2017 and were based on secondary structure features and machine learning algorithms, including the RNAcon tool, which was the reference classifier. Three steps are the basis of the proposed method:

1. The prediction of ncRNAs secondary structures, the extraction of frequent sub-structures as features and the classification of known ncRNA classes. To implement these processes, IPknot [13] algorithm was used to predict RNA secondary structures with pseudoknots,
2. the MoSS [29] decision tree pruning algorithm to obtain sub-structures
3. a deep learning network architecture, namely a convolutional neural network, as a supervised classifier.

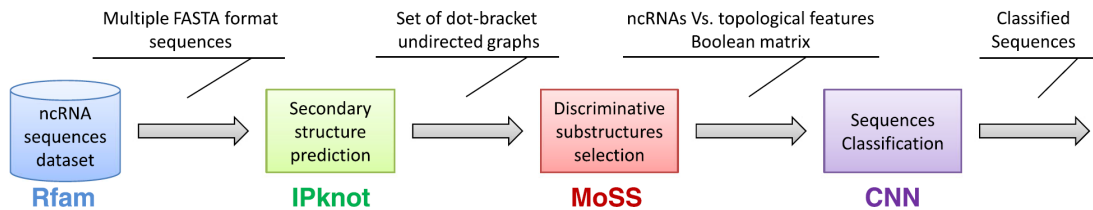


Figure 11: Pipeline of the nRC ncRNA sequence classification tool

5.3 GraPPLE

This study [30] investigates the use of specific properties of graphs that represent the predicted secondary structure of non-coding RNA (ncRNA) to reflect functional information. The authors developed a computational algorithm and a web-based tool called GraPPLE for classifying ncRNA molecules as functional and into Rfam families based on their graph properties. The tool was demonstrated to be more robust than sequence-similarity-based methods and covariance models with increasing sequence divergence and, when combined with existing methods, led to a significant improvement in prediction accuracy. The most informative graph properties were found to provide insight into the structural features that give ncRNA molecules functional properties. The GraPPLE tool may be useful for identifying potentially interesting ncRNA molecules among large candidate datasets.

5.4 ncRFP

Another approach [3], describes a method for predicting the family of non-coding RNAs (ncRNAs) called ncRFP. Traditional methods for ncRNA prediction involve predicting the secondary structure of the RNA and then identifying the ncRNA family based on the properties of the secondary structure. However, these methods can be complex and may not always be accurate due to errors that can accumulate in the multi-step process, particularly due to imperfections in tools used to predict the secondary structure of RNA. The ncRFP method is a novel approach that uses deep learning to predict the ncRNA family directly from the RNA sequence, bypassing the need to predict the secondary structure. This method simplifies the prediction process and improves accuracy compared to traditional methods. The neural network architecture of the AI model is shown in Figure 12

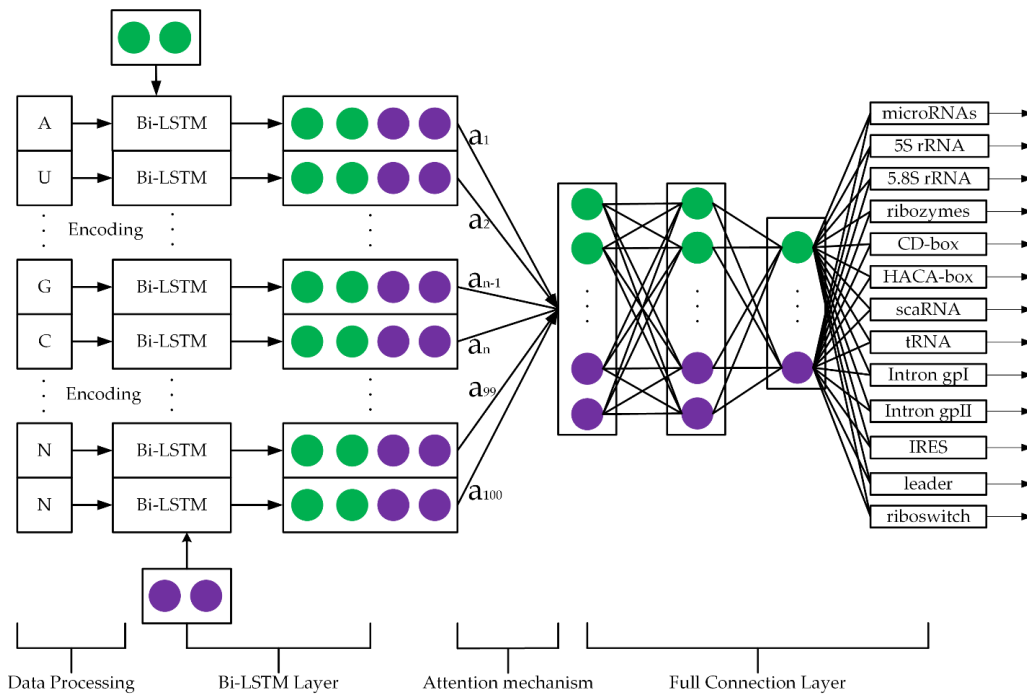


Figure 12: ncRFP NN model Architecture (Image Source [3])

5.5 ncDLRES

The authors of this article [31] propose a novel method called ncDLRES for predicting the family of non-coding RNA (ncRNA) based on dynamic long short-term memory (LSTM) and residual neural network (ResNet). The method, called ncDLRES, extracts the features of ncRNA sequences using dynamic LSTM and then classifies them using ResNet. The authors compare ncDLRES to both the homologous sequence alignment method and other methods that predict ncRNA based on secondary structure, called ncRFP. The homologous sequence alignment method is currently the most accurate method, but it has limitations due to the need for consensus secondary structure annotation of ncRNA sequences and the inability to

model pseudoknots. ncDLRES reduces the data requirements and expands the application scope compared to the homologous sequence alignment method, and its performance is greatly improved compared to the ncRFP methods.

5.6 ncDENSE

In this study [32], a method called ncDENSE, based on a deep learning model, was introduced. It predicts families of non-coding RNAs (ncRNAs) by analyzing the sequence features of these ncRNAs. The nucleotide bases in the sequences of ncRNAs were encoded using a one-hot coding scheme. These encoded sequences were then input into an ensemble deep learning model, which comprised three components: the dynamic bi-directional gated recurrent unit (Bi-GRU), the dense convolutional network (DenseNet), and the Attention Mechanism (AM). More specifically, the dynamic Bi-GRU was utilized to extract contextual feature information and capture long-term dependencies within the ncRNAs sequences. The AM was employed to assign varying weights to the features extracted by the Bi-GRU, focusing attention on information with higher weights. Meanwhile, DenseNet was employed to extract local feature information from the ncRNAs sequences and carry out classification using the fully connected layer.

6 ncRNA data

This section introduces various RNA databases, with a particular emphasis on the detailed utilization of the Rfam database. Additionally, it discusses the development of a specialized non-coding RNA database. This database is specifically designed for the training and testing of the proposed classifier. The section also provides a thorough analysis of the datasets employed, offering insights into their composition and utility in the context of RNA-based research and classification.

6.1 ncRNA databases

A non coding RNA database refers to a collection of sequences of ncRNA, along with their corresponding structural explanations. These databases serve as a tool for researchers to explore the world of ncRNAs by accessing and analyzing extensive amounts of reliable data. Some of these databases often include multiple sequence alignments (MSAs) of ncRNA families and functional and structural annotations for each ncRNA family. The below list presents a selection of coding RNA (ncRNA) databases.

- Rfam [8] A comprehensive database of ncRNA families and their annotated alignments, curated by the RNA Bioinformatics Group at the University of Cambridge.
- RNACentral [10] Similar to Rfam, RNACentral acts as a unified hub, providing centralized access to non-coding RNA sequences gathered from a network of specialized databases.
- IRESbase [33] A freely accessible online database that curates experimentally validated internal ribosome entry sites (IRESs)
- miRBase [34] A database of microRNAs (miRNAs), which are small ncRNAs that play a role in the regulation of gene expression.
- NONCODE [35] is an integrated knowledge database dedicated to non-coding RNAs (excluding tRNAs and rRNAs).
- Ensembl [36] is a genome browser for vertebrate genomes and model organisms that supports research in comparative genomics, evolution, sequence variation and transcriptional regulation
- GeneCards [37] is a searchable, integrative database that provides comprehensive, user-friendly information on all annotated and predicted human genes
- LNCipedia [38] is a public database for long non-coding RNA (lncRNA) sequence and annotation.

6.2 RNACentral

RNACentral [10] is a comprehensive, publicly accessible resource that provides integrated access to a vast array of non-coding RNA (ncRNA) sequences. These sequences are sourced from a collaboration of over 54 expert databases, covering a wide range of organisms and RNA types. The platform is coordinated by the European Bioinformatics Institute [39] and supported by wellcome.com, with initial funding from BBSRC



Figure 13: RNACentral Expert Databases [Image source]

RNACentral Support:

- **Data Integration:** It compiles ncRNA sequences from multiple databases, offering tools for text search, sequence similarity search, bulk downloads, and programmatic data access
- **Stable Identifiers:** Unique identifiers are assigned to each distinct sequence, with support for species-specific identifiers for sequences in specific organisms
- **Genomic Mapping:** Sequences are mapped to reference genomes from over 250 species using blat, and users can browse these mapped sequences in a genome browser or download genome coordinates in various formats
- **Functional Annotations:** RNACentral incorporates modified nucleotides from sources like Modomics and PDB, miRNA targets from TarBase, and Gene Ontology annotations from QuickGO. All sequences are annotated with

Rfam models, providing warnings and additional information, including secondary structure diagrams for various RNA types

6.3 IRESbase

IRESbase [33] is a publicly accessible repository of experimentally validated IRES sequences from diverse organisms, including eukaryotes and viruses. Curated by researchers from Nanjing University of Aeronautics and Astronautics, IRESbase currently holds over 1,300 IRES entries, encompassing both eukaryotic and viral IRESs from 11 eukaryotic species and 198 viruses, respectively. The database meticulously annotates each IRES record, providing detailed information on its location, host RNA, functional properties, and supporting literature references.

IRESbase offers a wealth of features that enhance its utility for research purposes. Users can search for IRESs based on various criteria, including organism, host RNA type, functional attributes, and sequence similarity. Additionally, the database provides a comprehensive mapping of IRESs to human circular RNAs (circRNAs) and long non-coding RNAs (lncRNAs), highlighting their potential involvement in these emerging RNA classes.

The accessibility and comprehensiveness of IRESbase have propelled its use in numerous research endeavors. Studies have employed IRESbase data to investigate the evolution, structural features, and functional mechanisms of IRESs, contributing to a deeper understanding of their role in gene regulation. Additionally, IRESbase has been instrumental in predicting and analyzing novel IRESs, facilitating the identification of previously uncharacterized IRES-dependent genes.

6.4 Rfam

Rfam [8] is a database of non-coding RNA families and their annotated alignments, curated by the RNA Bioinformatics Group at the University of Cambridge. It is a comprehensive resource for ncRNA sequences and their corresponding functional and structural annotations.

The Rfam database relies on sets of related sequences, known as families, which share a common ancestor and are thought to have similar functions. These families are identified through sequence alignments (MSAs) that are generated using tools and meticulously checked by knowledgeable annotators to guarantee their reliability and precision. Besides the MSAs Rfam also provides a variety of annotations for each ncRNA family, including functional descriptions, secondary structure predictions, and cross-references to other databases.

Rfam serves as a tool, for researchers delving into the study of ncRNAs. It offers access to reliable data, enabling analysis on a large scale. One of its benefits lies in aiding the identification and categorization of ncRNA sequences, providing an extensive collection of well established ncRNA families for reference purposes. Rfam remains consistently updated with additions of families and annotations, rendering it an indispensable resource, within the RNA community.

6.4.1 Rfam MySQL database

Rfam provides a public read-only MySQL database containing the latest version of Rfam data. Details to access this database are provided in Table 1. The database core scheme is shown in diagram 14.

Parameter	Value
host	mysql-xfam-public.ebi.ac.uk
port	4497
user	xfamro
database	Rfam

Table 1: Rfam MySQL Database - Connection Details

Advanced examples of using the public Rfam database can be found in Current Protocols in Bioinformatics publication [40]. Some examples bellow:

```
1
2  -- Number of Sequences per file and ncRNA type
3  -----
4  SELECT family.rfam_acc , family.type ,
5         COUNT(full_region.rfamseq_acc) as Number_of_Sequences
6  FROM full_region, family
7  Where family.rfam_acc = full_region.rfam_acc
8  GROUP BY family.rfam_acc;
9
10 -- For RF00014.fasta file show
11 -- sequence id, start and stop, description and type
12 -- This query will be used to construct fasta headers
13 -----
14 SELECT full_region.rfam_acc, full_region.rfamseq_acc, seq_start, seq_end,
15        rfamseq.description, family.type
16 FROM full_region , rfamseq , family
17 WHERE full_region.rfamseq_acc = rfamseq.rfamseq_acc
18 AND family.rfam_acc = full_region.rfam_acc
19 AND full_region.rfam_acc = 'RF00014';
20
```

6.5 Data Collection

To assemble a dataset of small non-coding RNA sequences, Rfam database was mainly used, The bellow SQL query was used to retrieve all .fasta file names that contain RNA sequences of the RNA families in interest.

```
1  -- Get all .fasta file names
2  -----
3  SELECT rfam_acc FROM family
4  WHERE type = "Cis-reg; IRES;" OR type = "Cis-reg; leader;"
5  OR type = "Cis-reg; riboswitch;" OR type = "Gene; miRNA;"
```

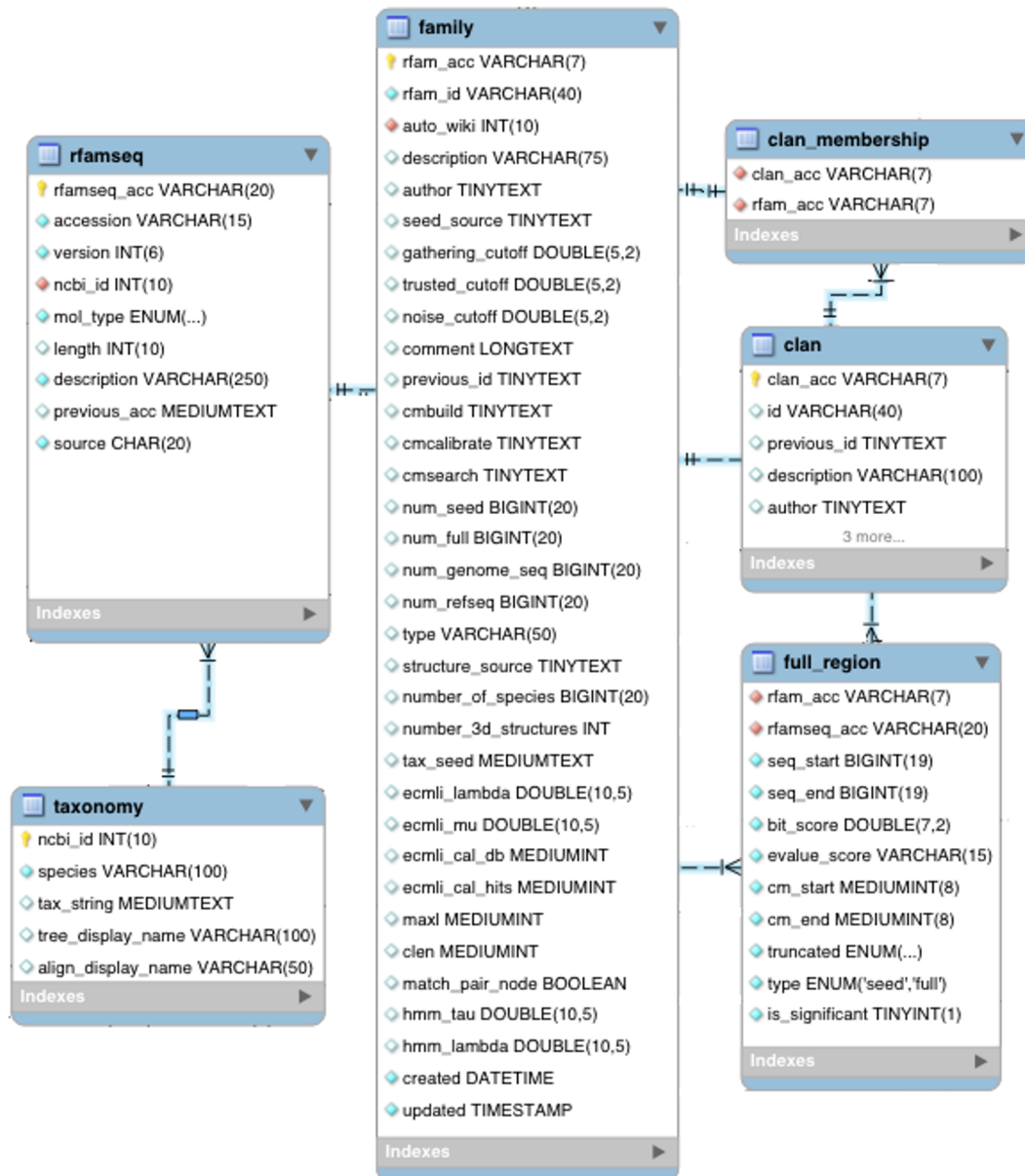


Figure 14: Rfam core database scheme diagram [Image Source]

```

6 OR type = "Gene; ribozyme;" OR type = "Gene; snRNA; snoRNA; CD-box;"
7 OR type = "Gene; snRNA; snoRNA; HACA-box;" OR type = "Gene; snRNA; snoRNA; scaRNA;"
8 OR type = "Gene; tRNA"

```

This query returns a list of all the fasta filenames:

```

1 # The response of the MySQL rfam server
2 # -----
3 ['RF00008', 'RF00009', 'RF00010', 'RF00011', ... , 'RF04235', 'RF04236']

```

Each file is downloaded from the SFTP server :

```

1 acc_id_list = ['RF00008', 'RF00009', 'RF00010', 'RF00011' , 'RF04235', 'RF04236']
2 for filename in acc_id_list:
3     URL = "http://http.ebi.ac.uk/pub/databases/Rfam/CURRENT/fasta_files/"
4     URL += filename + ".fa.gz"
5     wget.download(URL, "../datasets/Rfam/"+filename+".fa.gz")

```

Example of RF00004.fa fasta file contents:

```

1 >CM001883.1/36104473-36104278 Theobroma cacao cultivar Matina 1-6 chromosome ...
2 ATACCTTTCTCGGCCTTTTGGCTAAGATCAAGTGTAGTATCTGTTCTTATCAGTTAATATCTGATACGTGGGCCA ...
3 >JH795869.1/919604-919793 Dacryopinax sp. DJM-731 SS1 chromosome Unknown DAC ...
4 GCACCACTCTGGCCTTTTGGCTTAGATCAAGTGTAGTATCTGTTCTTATTAGTTTAACCACTAATATGGTCGCACC ...
5 >CM001769.1/9270458-9270652 Cicer arietinum chromosome Ca6, whole genome sho ...
6 ATACCTTTCTCGGCCTTTTGGCTAAGATCAAGTGTAGTATCTGTTTTTATCAGTTAATATCTGATATGTGGTCCA ...
7 >FR853084.2/62466812-62466957 Gorilla gorilla gorilla genomic chromosome, ch ...
8 ATTACTTCTCAGCCTTTTGGCTAAGATCAAGTGAATAAATCTCATTGTGCTTTATGCCTAATGTGTGCTTATATT ...
9 >KK088422.1/566554-566746 Aspergillus ruber CBS 135680 unplaced genomic scaf ...
10 CCAGCTCTCTTTGCCTTTTGGCTTAGATCAAGTGTAGTATCTGTTCTTTTCAGTTAATCTCTGAAAGTGTCTAA ...
11 >AACT01051284.1/529-401 Ciona savignyi cont_51284, whole genome shotgun sequ ...
12 ACAGCTGATGCCGAGCTACACTATGTATTAATCGGATTTTGAACCTGGAGTACGGTTCTGGAGCTTGCTCCACC ...

```

6.5.1 Fasta files

The FASTA file format is frequently employed as a text based representation, for sequences encompassing nucleotide sequences (DNA or RNA) as well as amino acid sequences (proteins). It serves as an adaptable format that is widely utilized in the fields of bioinformatics and biochemical research. A FASTA file comprises sequence records, with each record consisting of two components:

1. Definition Line: The first line of a sequence record starts with a greater-than sign (" $>$ ") followed by a unique identifier for the sequence. This identifier can be any descriptive name or code, such as the name of the organism or the source of the sequence.
2. Sequence Data: The subsequent lines of a sequence record contain the actual sequence data. The sequence data is composed of single-letter codes representing the individual nucleotides or amino acids in the sequence.

Since this research has classification interests only, in "definition" section of each sequence the class will be the only available info, as shown in the below example.

```

1 >IRES
2 ATACCTTTCTCGGCCTTTTGGCTAAGATCAAGTGTAGTATCTGTTCTTATCAGTTAATATCTGATACGTGGGCCA ...
3 >tRNA
4 GCACCACTCTGGCCTTTTGGCTTAGATCAAGTGTAGTATCTGTTCTTATTAGTTTAACCACTAATATGGTCGCACC ...
5 >tRNA
6 ATACCTTTCTCGGCCTTTTGGCTAAGATCAAGTGTAGTATCTGTTTTTATCAGTTAATATCTGATATGTGGTCCA ...
7 >riboswitch
8 ATTACTTCTCAGCCTTTTGGCTAAGATCAAGTGAATAAATCTCATTGTGCTTTATGCCTAATGTGTGCTTATATT ...

```

```

9 >HACA-box
10 CCAGCTCTCTTTGCCTTTTGGCTTAGATCAAGTGTAGTATCTGTTCTTTTCAGTTAATCTCTGAAAGTGTCTAA ...
11 >tRNA
12 ACAGCTGATGCCGCAGCTACACTATGTATTAATCGGATTTTGAACCTGGAGTACGGTTCTGGAGCTTGCTCCACC ...

```

6.6 Final dataset (NCC dataset)

Files of the same family were combined in order to generate one .fasta file per RNA family. IRES non-coding RNA family dataset was poor, so a second data source, IRESbase [33], dedicated to this family was used to extend the initial dataset. In table 2 the number of sequences of each family:

RNA Family	Source	# of Sequences
IRES	Rfam	1472
IRES	IRESbase	1328
leader	Rfam	31662
riboswitch	Rfam	69465
miRNA	Rfam	387173
ribozyme	Rfam	220007
CD-box	Rfam	132915
HACA-box	Rfam	36938
scaRNA	Rfam	2962
tRNA	Rfam	1432442
5S_rRNA	Rfam	140644
5_8S_rRNA	Rfam	4940
Intron_gpI	Rfam	2611
Intron_gpII	Rfam	15729

Table 2: Number of Sequences per family per Data source

The final dataset has nearly 4-5 thousand sequences per family, with some exceptions. The distribution of our data is shown in Figure 15, and the number of sequences per class is shown in Table 3

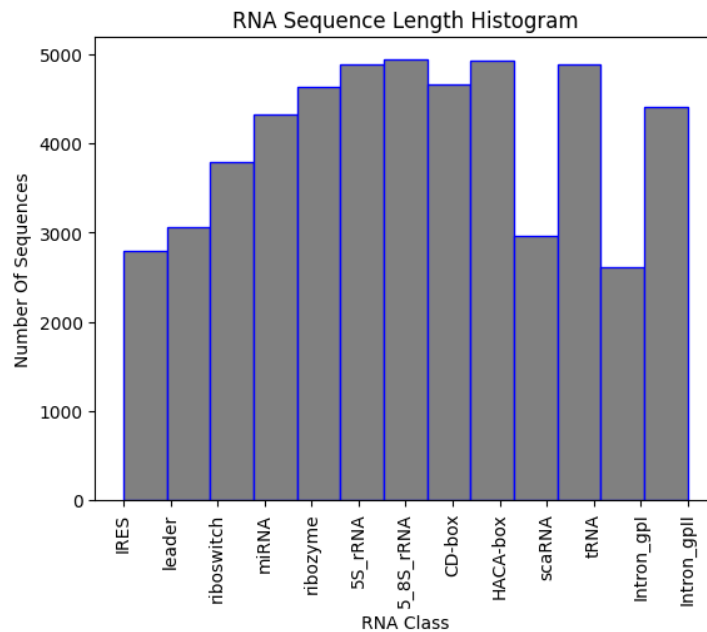


Figure 15: Number of sequences per class distribution of final dataset (NCC dataset)

RNA Family	Number of Sequences
IRES	2800
leader	3061
riboswitch	3791
miRNA	4317
ribozyme	4630
CD-box	4661
HACA-box	4931
scaRNA	2962
tRNA	4882
5S_rRNA	4882
5_8S_rRNA	4940
Intron_gpI	2611
Intron_gpII	4409

Table 3: Number of Sequences per family in NCC dataset

The analysis of the distribution of sequence lengths, among RNA classes as shown in Figure 17 indicates a relationship between the length of a sequence and its corresponding RNA class. This correlation has the potential to enhance the effectiveness of classification models. For example, as illustrated in Figure 16 miRNA sequences are generally shorter than 200 nucleotides while ribozyme RNA sequences often exceed 800 nucleotides in length. These distinct length characteristics within RNA classes suggest that considering sequence length could be valuable for distinguishing between RNA classes in classification models. This understanding of the varying lengths of RNA sequences across classes not only deepens our knowledge about RNA structure but also paves the way for developing more accurate and efficient

techniques, for classifying RNAs.

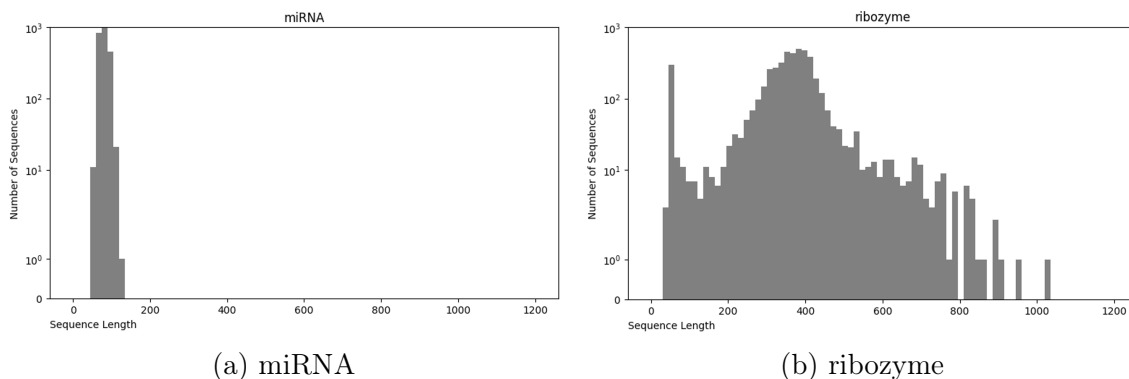


Figure 16: Sequence length distribution of miRNA and ribozyme

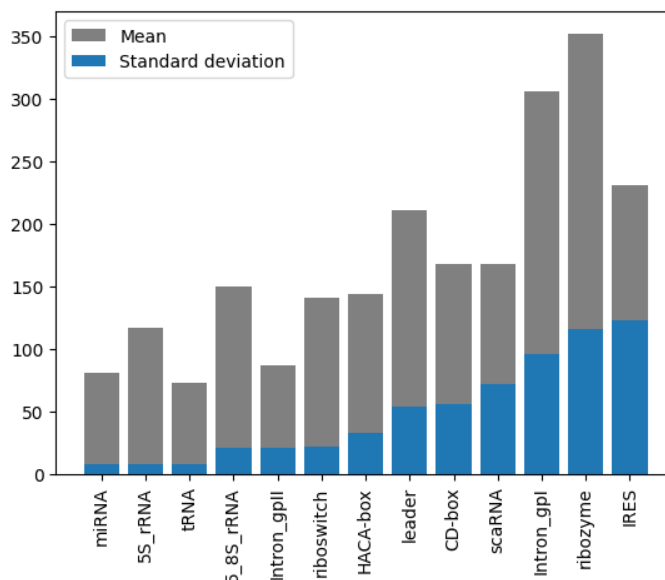


Figure 17: Sequence length mean and STB, per class, sorted by STB of NCC dataset

6.7 nRC dataset

The dataset used in the nRC [4] publication was specifically designed for the classification of non-coding RNA (ncRNA) sequences. This dataset was constructed by utilizing sequences from the Rfam database. The selected dataset is composed by 13 different ncRNA classes: miRNA, 5S rRNA, 5.8S rRNA, ribozymes, CD-box, HACA-box, scaRNA, tRNA, Intron gpI, Intron gpII, IRES, leader, and riboswitch.

To create a balanced and statistically meaningful dataset, the researchers followed a methodology similar to that used in previous studies. They selected 20% non-redundant sequences using the CD-HIT tool [41], which is a widely used method for clustering and comparing protein or nucleotide sequences. The aim was to ensure that the dataset did not have redundant sequences which might skew the classification results. For most of the ncRNA classes, they randomly chose 500 sequences

each. However, for the IRES class, only 320 sequences were available, leading to a total of 6320 ncRNA sequences in the dataset. A second dataset composed of 2600 sequences, downloaded by Rfam for testing purposes of the nRC tool, containing 200 sequences of each class.

Train	6320 seq
Test	2600 seq

Table 4: nRC [4] dataset divided to Train and Test

Given that this dataset is already in use for benchmarking with other classification models, we will also employ it for our proposed model’s training and testing to enable a fair comparison.

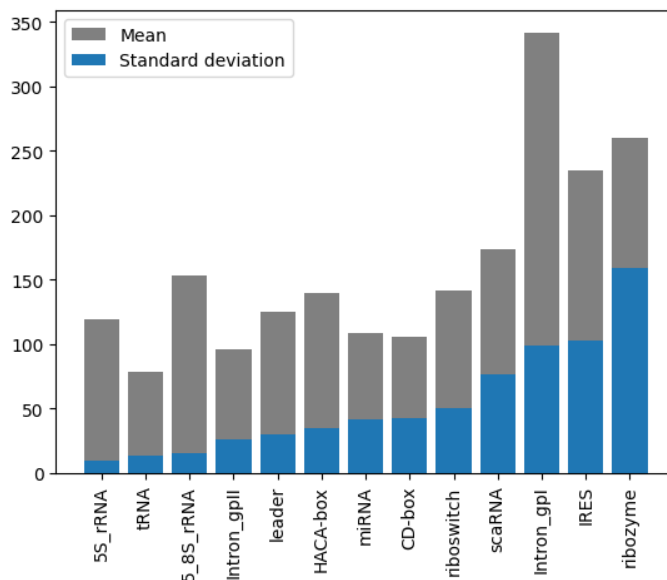
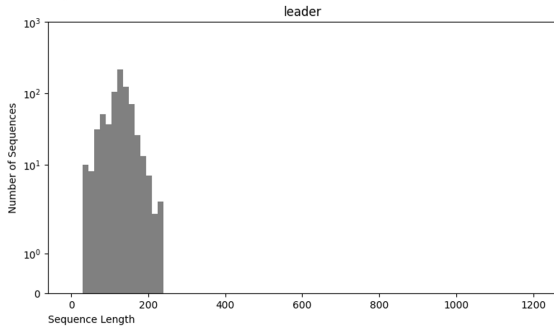


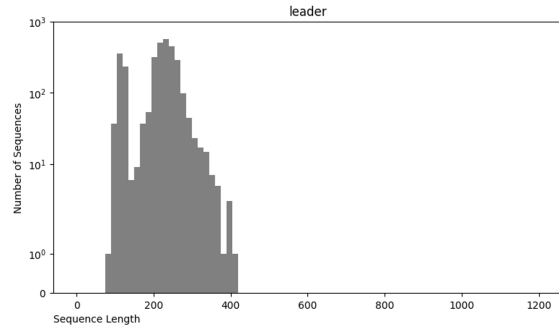
Figure 18: Sequence length mean and STB, per class, sorted by STB of nRC dataset

6.7.1 Comparison between NCC and nRC datasets

The Non-Coding RNA Classification (NCC) dataset developed for this research is significantly more extensive, containing over ten times the number of sequences compared to previous nRC dataset. Interestingly, the distribution of RNA sequence lengths across different RNA classes in this new dataset is similar to that observed in nRC dataset. This similarity is particularly evident in certain RNA classes. For instance, IRES sequences, ribozymes, and Intron_gpI all tend to be longer in both the NCC and nRC datasets, although there are some minor variations, as presented in Figures 20 and 21. A noteworthy distinction can be observed in the case of leader RNA sequences. This difference is clearly illustrated in Figure 19, where the distribution of the leader RNA class sequence lengths in the datasets are compared side by side.

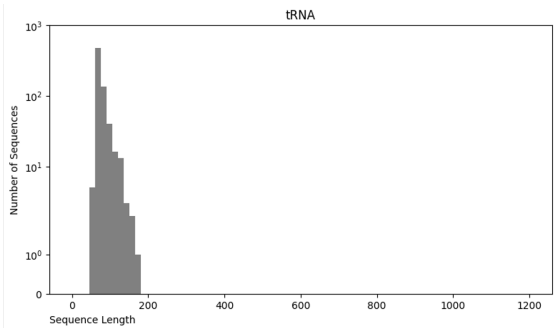


(a) leader in nRC

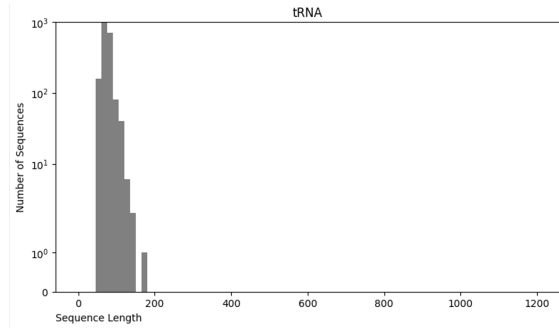


(b) leader in NCC

Figure 19: Sequence length distribution of leader RNA class in nRC (a) and NCC (b) dataset

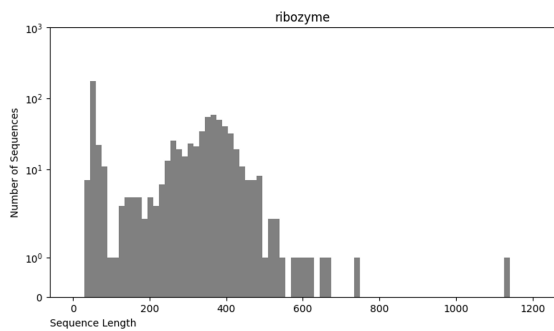


(a) tRNA in nRC

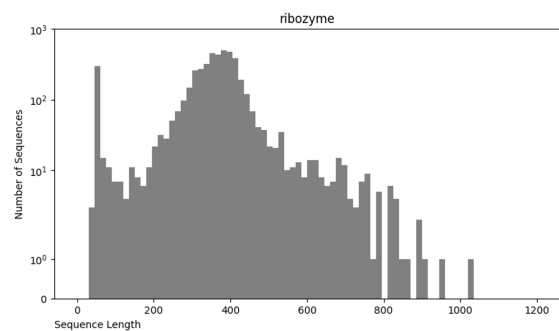


(b) tRNA in NCC

Figure 20: Sequence length distribution of tRNA RNA class in nRC (a) and NCC (b) dataset



(a) ribozyme in nRC



(b) ribozyme in NCC

Figure 21: Sequence length distribution of ribozyme RNA class in nRC (a) and NCC (b) dataset

7 Implementing the Prediction Mechanism.

In this section, the document focuses on implementing the prediction mechanism for the non-coding RNA (ncRNA) classification model. The section is divided into several parts, detailing the steps taken in preparing the data, developing the model architecture, and training and evaluating the model.

The purpose of this research is the development of a non-coding RNA classifier using as input the RNA sequence (primary structure of RNA). Focusing only on primary structure of RNA, error from prediction methods of secondary structure (graph properties of RNA) are excluded from the classification flow. It is crucial to focus also on other features of RNA like secondary structure or k-mers substrings, but it is out of the scope of this research.

So to summarize, the main goal of this research focus on are developing a non-coding RNA classifier the bellow characteristics:

1. Input of the model is the RNA sequence (primary structure) only.
2. The architecture of the model should be very simple, like ncRFP [3], resulting in quick training sessions.
3. Acceptable prediction accuracy of the model.

7.1 Data preparation

Data preparation consist of tow main parts, first the processes of sequence padding and cutting to standardize RNA sequence lengths to 500 nucleotides, ensuring consistency crucial for the model's accuracy. The second part is the implementation of one-hot encoding, a technique used to transform RNA sequences into a numerical format that is more suitable for machine learning algorithms.

7.1.1 Sequence Padding and Cutting

Padding and cutting are common techniques used to prepare sequences for deep learning models, particularly in sequential data tasks such as natural language processing (NLP) and speech recognition. These techniques are employed to ensure that all sequences have a consistent length, which can improve the performance of the model.

Padding involves adding extra elements to shorter sequences to make them match the length of the longest sequence. This is typically done with zeros or filler values. Padding ensures that all sequences receive equal attention from the model, regardless of their original length. It also helps to prevent the model from overfitting to shorter sequences, as it is forced to learn features from all sequences, regardless of their length.

Cutting involves removing elements from longer sequences to make them match the length of the shortest sequence. This is typically done by removing the last elements from the sequence. Cutting ensures that all sequences fit within the memory constraints of the model and the computational resources available. It also helps to prevent the model from becoming too complex and difficult to train.

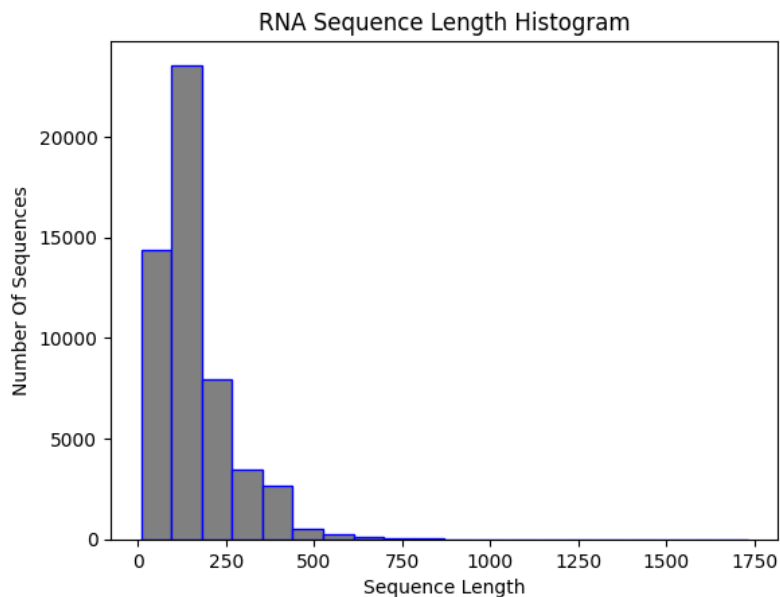


Figure 22: NCC dataset, RNA Sequences length distribution

By inspecting Figure 22 it is notable that most sequences have less than 500 nucleotides. So for this model, the input sequence must have length equal to 500 features. That means that each sequence with fewer nucleotides will be padded with zeroes and if the sequence has greater length it will be cutted, risking losing critical features of the sequence.

7.1.2 One-hot encoding

One-Hot Encoding is a process used in data preprocessing to convert categorical data into a numerical format, making it suitable for use in machine learning algorithms which typically require numerical input. This technique involves creating a binary vector for each category in the dataset.

The primary advantage of One-Hot Encoding is that it avoids the introduction of a false ordinal relationship that might occur if the categories were simply encoded as integers (e.g., 1 for Red, 2 for Blue, 3 for Green). In the integer format, algorithms might misinterpret the data to imply that Green is greater than Blue, and Blue is greater than Red, which is not the case. By using binary vectors, One-Hot Encoding maintains the distinctiveness of each category without implying any order.

However, One-Hot Encoding can significantly increase the dimensionality of the dataset, especially if the categorical variable has many categories. This increase in dimensions can lead to computational complexity and the risk of overfitting in machine learning models. Therefore, it is often used judiciously and sometimes in conjunction with dimensionality reduction techniques.

Since there are no known relations between the bases of RNA One-hot encoding was used to encode the sequences before using them as input into the deep learning model. There are 16 different characters that we need to represent as shown in table 6, but most of the characters are not present in the dataset and if they do, they are very rare. Therefore, the encoding is applied to the basic RNA known bases

and the extra IUPAC characters are encoded with 'X' which is also the padding character. Table 5 shows the different encodings of RNA characters, encoded with 4 and 8 digits.

Base	4 digits	8 digits
A	1000	1000 0010
T/U	0100	0100 0001
G	0010	0010 1000
C	0001	0001 0100
X	0000	0000 0000

Table 5: One hot encoding of RNA Bases

The main reason Table 5 was chosen instead of complete IUPAC encoding of 16 characters was because both NCC and nRC datasets contains very few of the rest set presented in Table 6.

IUPAC Code	Meaning	Complement	Encoding
A	A	T	1000 0000 0000 0000
C	C	G	0100 0000 0000 0000
G	G	C	0010 0000 0000 0000
T/U	T	A	0001 0000 0000 0000
M	A or C	K	0000 1000 0000 0000
R	A or G	Y	0000 0100 0000 0000
W	A or T	W	0000 0010 0000 0000
S	C or G	S	0000 0001 0000 0000
Y	C or T	R	0000 0000 1000 0000
K	G or T	M	0000 0000 0100 0000
V	A or C or G	B	0000 0000 0010 0000
H	A or C or T	D	0000 0000 0001 0000
D	A or G or T	H	0000 0000 0000 1000
B	C or G or T	V	0000 0000 0000 0100
N	G or A or T or C	N	0000 0000 0000 0010
X	None	-	0000 0000 0000 0000

Table 6: One hot encoding of RNA IUPAC

7.2 NCC Model Architecture

Over 25 distinct models were developed, trained, and tested using the NCC dataset. This extensive experimentation included dense networks and convolutional networks, both independently and in combined sequential formats. However, it was the recurrent models that exhibited exceptional performance in terms of accuracy, outshining the other network types. Notably, bidirectional recurrent neural networks (BiRNNs) stood out.

In RNA classification, a Bidirectional Recurrent Neural Network layer is particularly effective due to its ability to process sequential data in both forward and backward directions. This bidirectional approach is crucial because the context of RNA sequences often depends on both preceding and succeeding nucleotides. Traditional RNNs, which process data in a single direction, might miss important context or patterns that are only apparent when considering the full sequence. A BiRNN can capture these dependencies by analyzing the sequence from both ends, leading to a more comprehensive understanding of the RNA structure and function. This enhanced ability to recognize patterns and relationships within RNA sequences makes BiRNNs especially suitable for tasks like RNA classification, where the sequential context and order are critical for accurate predictions.

The best model, which exhibited the highest accuracy and acceptable loss margins while maintaining its simplicity of its architecture, was selected and presented in this research. It consists of one-dimensional convolutional neural network, one-dimensional max pooling layer, bidirectional recurrent layer and one fully connected dense layer. The model architecture is illustrated in Figure 23.

This research prioritizes not only accuracy but also the speed of training and testing periods. The use of a straightforward architecture facilitates these quick processes. By simplifying the system's design, we can efficiently train and test our models, achieving rapid results without compromising on accuracy. This dual focus on speed and precision is a core objective of our study.

1. Input Layer
2. Convolutional Layer
3. Max Pooling Layer
4. Bidirectional Recurrent Layer
5. Fully Connected Dense Network

The convolutional layer extracts important features from the input data, while the max pooling layer downsamples the feature map to reduce its size. The bidirectional recurrent layer captures long-term dependencies in the input data, while the fully connected dense layer combines the features extracted by the previous layers to make a final prediction.

This model architecture was shown to achieve the highest accuracy and acceptable loss margins on the NCC dataset, making it a promising choice for a variety of sequential data classification tasks.

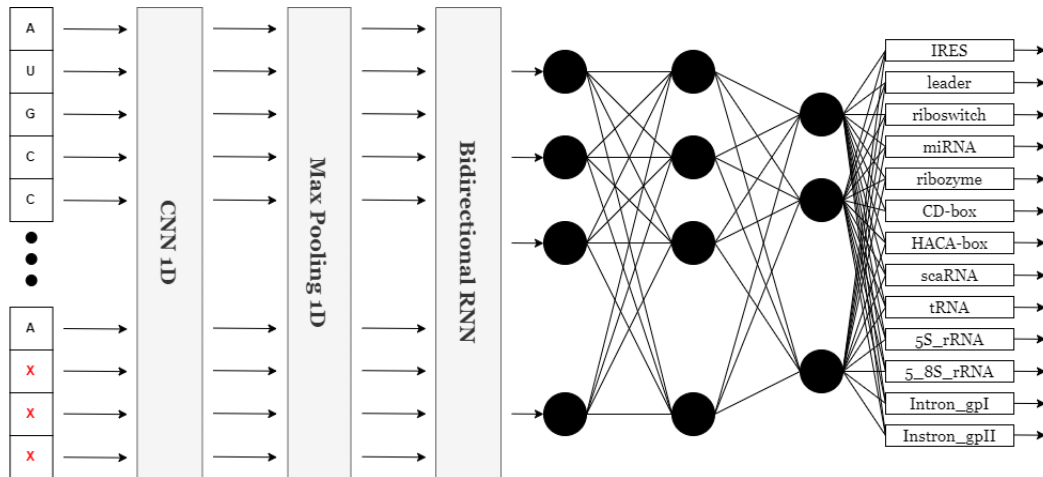


Figure 23: NCC Neural Network Architecture

This Keras model is a neural network for processing sequential data for a classification task with 13 classes (as indicated by the final Dense layer with 13 units and 'softmax' activation). Breakdown of its components with more details:

1. **Input Layer:** Accepts input with shape (500, 4) or (500, 8) where 500 is the length of padded and cutted sequence and 4 or 8 is the number of digits used for one hot encoding, in general this is indicating sequences of length 500 with 4 or 8 features each.
2. **Conv1D Layer:** A one-dimensional convolutional layer with 32 filters, kernel size 9, and ReLU activation. It's designed to extract features from the sequence data.
3. **MaxPooling1D Layer:** Reduces the dimensionality of the data, pooling over windows of size 4, to condense the features and reduce computation.
4. **Bidirectional GRU Layer:** A bidirectional GRU (Gated Recurrent Unit) with 128 units. It processes the data in both forward and backward directions, capturing dependencies in the sequence. It includes dropout equal to 0.3 (for regularization) and random kernel and recurrent initializers, while zero initialization of bias.
5. **Flatten Layer:** Flattens the output of the GRU layer to a single dimension, preparing it for the dense layer.
6. **Dense Layers:** Fully connected layers with the last dense layer composed of 13 units and 'softmax' activation, outputting a probability distribution over 13 classes.

7.3 Training and Testing the Model

Adam optimization method [42] was used with learning rate 0.001. The Adam optimizer is a widely-used optimization algorithm in training neural networks. It combines the best properties of the Stochastic Gradient Descent and RMSProp algorithms to handle sparse gradients on noisy problems. Adam stands for "Adaptive Moment Estimation," and utilizes the squared gradients for scaling the learning rate, similar to RMSprop, and employs the moving average of the gradient, akin to SGD with momentum, rather than the gradient itself. This integration effectively marries the concepts of dynamic learning rate adjustment and gradient smoothening. Such a combination is key in efficiently navigating towards the global minimum in the optimization landscape. This approach helps in navigating the rough terrain of high-dimensional space, which is typical in deep learning tasks. Adam is favored for its efficiency in terms of memory requirement, computational cost, and especially for its effectiveness in cases where relatively large amounts of data and parameters are involved.

Categorical Cross-Entropy is a loss function often used in machine learning for multi-class classification problems. It measures the difference between two probability distributions - the true distribution (actual labels) and the predicted distribution, output by the model. The function calculates the loss by taking the negative log of the predicted probability assigned to the true class. It's particularly effective when the model's output is a probability distribution across multiple classes, like in neural networks with softmax activation in the output layer. This loss function is crucial for models where accurate probability distributions are essential, such as in classification tasks with more than two classes.

$$L = - \sum_{i=1}^M y_i \log(p_i) \quad (11)$$

And for multi-class classification:

$$L = - \sum_{i=1}^M y_{o,i} \log(p_{o,i}) \quad (12)$$

Where:

- L is the loss.
- M is the number of classes.
- y_i is a binary indicator (0 or 1) if class label i is the correct classification for the observation.
- p_i is the predicted probability of the observation belonging to class i .
- $y_{o,i}$ is a binary indicator (0 or 1) if class i is the correct classification for observation o .
- $p_{o,i}$ is the predicted probability of observation o being of class i .

The training of the model was conducted over three different sets of epochs: 20, 50, and 100. According to the data illustrated in Figure 24, it was observed that

beyond the first 20 epochs, the model’s accuracy plateaued, showing little to no further improvement, and this trend was mirrored in the loss metrics as well. All the details about the model’s training are shown in Table 7.

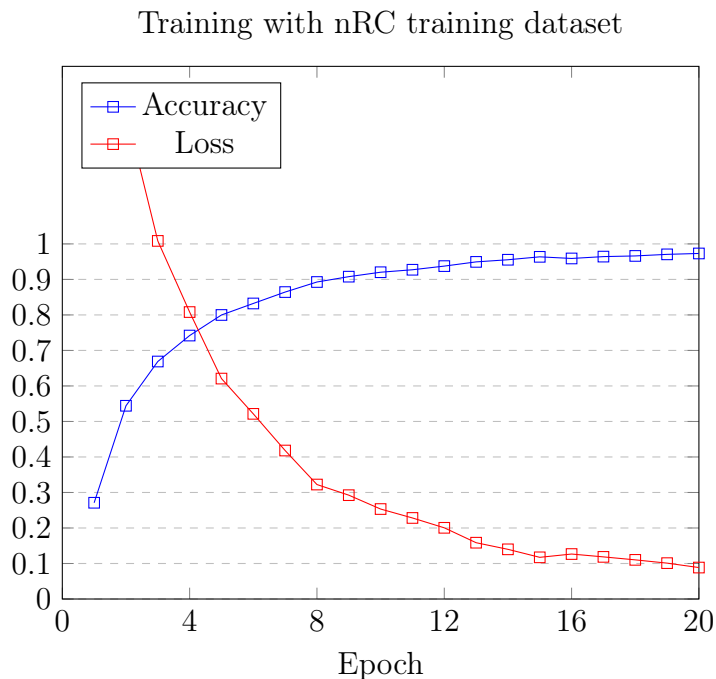


Figure 24: Diagram of Accuracy and loss per epoch during training

Number of Epochs	20/50/100
Batch size	128
Steps per Epoch	Default = 277
Optimizer	Adam
loss function	Categorical Cross Entropy
Shuffle Train data	True

Table 7: NCC model training parameters

7.3.1 Performance metrics

To assess the performance of each non-coding RNA class prediction method, mentioned in this document, the well-established metrics of accuracy, sensitivity, precision, F1-score, and Matthews correlation coefficient (MCC) were employed as well as the confusion matrix.

- **Accuracy** is the percentage of correctly classified images.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FN} + \text{FP}} \quad (13)$$

- **Sensitivity**, also known as **Recall**, is the ratio of correctly predicted positive cases to the whole actual positive cases.

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (14)$$

- **Precision** is the ratio of correctly predicted positive cases to the total predicted positive cases.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (15)$$

- The **F1-score** is a combination of the precision and recall metrics and can be defined as the harmonic mean of a model's precision and recall.

$$\text{F-score} = 2 \times \frac{\text{Precision} \times \text{Sensitivity}}{\text{Precision} + \text{Sensitivity}} \quad (16)$$

- The **Matthews correlation coefficient (MCC)** takes into account true and false positives and negatives, and is an efficient metric for unbalanced classes.

$$\text{MCC} = \frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}} \quad (17)$$

In case of multi-class classification, accuracy measures the proportion of correctly classified cases from the total number of objects in the dataset.

$$\text{Accuracy} = \frac{\text{Correct predictions}}{\text{All predictions}} \quad (18)$$

For the other metrics in case of multi-class classification first, measurement of the metric by class is applied, then macro-average it across classes. Example of precision is presented in equations 19 and 20.

$$\text{Precision}_{\text{ClassA}} = \frac{\text{TP}_{\text{ClassA}}}{\text{TP}_{\text{ClassA}} + \text{FP}_{\text{ClassA}}} \quad (19)$$

$$\text{Precision}_{\text{Macro-Average}} = \frac{\text{Precision}_{\text{ClassA}} + \dots + \text{Precision}_{\text{ClassN}}}{N} \quad (20)$$

In the multi-class the Matthews correlation coefficient (MCC) can be defined in terms of a confusion matrix C for K classes. To simplify the definition, consider the following intermediate variables:

- t_k : the number of times class k truly occurred

$$t_k = \sum_i^K C_{ik} \quad (21)$$

- p_k : the number of times class k was predicted

$$p_k = \sum_i^K C_{ki} \quad (22)$$

- c : the total number of samples correctly predicted

$$c = \sum_k^K C_{kk} \quad (23)$$

- s : the total number of samples

$$s = \sum_i^K \sum_j^K C_{ij} \quad (24)$$

Then the multiclass MCC is defined as:

$$MCC = \frac{c \times s - \sum_k^K p_k \times t_k}{\sqrt{(s^2 - \sum_k^K p_k^2) \times (s^2 - \sum_k^K t_k^2)}} \quad (25)$$

Confusion Matrix is a table used in classification tasks to visualize the performance of an algorithm. It displays the number of correct and incorrect predictions categorized by their actual and predicted classifications, typically with actual classes in rows and predicted classes in columns, helping to easily identify where the model is making mistakes. The most basic terms for a binary classifier, as presented in Figure 25, are True positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN).

- TP: values which are actually positive and are predicted positive
- TN: values which are actually negative and are predicted negative
- FP: values which are actually negative and are predicted positive
- FN: values which are actually positive and are predicted negative

Actual	Positive	TP	FN
	Negative	FP	TN
		Positive	Negative
		Predicted	

Figure 25: Confusion Matrix for binary classification problem

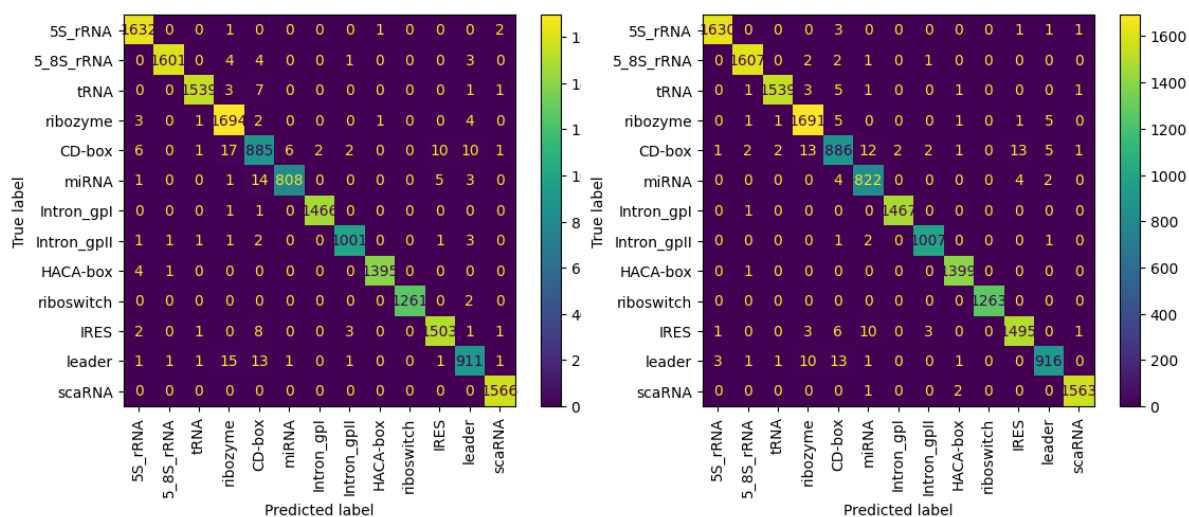
It is important to note that these basic metrics are extracted for each class separately, since the task at hand involves multi-class classification. For a single class (for example Class C_k), positive is considered a sample belonging to the specific class, while negative is considered a sample belonging to any other class. So, a confusion matrix for a multi class problem looks like Figure 26

		C_k	
	TN	FP	TN
Actual	FN	TP	FN
	TN	FP	TN
		C_k	
		Predicted	

Figure 26: Confusion Matrix for multi-class classification problem

7.3.2 Benchmarking NCC dataset

First step is splitting the NCC dataset to Train and Test with test size equal to 33% of the main dataset. Training the NCC model with 35427 number of sequences using the parameters from Table 7 and testing it with the rest 17450.



(a) 8 Features per RNA base

(b) 4 Features per RNA base

Figure 27: Confusion Matrix of NCC model trained and tested with NCC dataset matrix (a) is with 8 digits encoding and matrix (b) with 4 digits per RNA base

One-hot Enc	Accuracy	Sensitivity	Precision	F-Score	MCC
8 Digits	0.989226	0.986510	0.987894	0.987182	0.988283
4 Digits	0.990544	0.988647	0.988861	0.988732	0.989716

Table 8: Test metrics using the NCC dataset with 4 and 8 digits RNA Base encoding

7.3.3 Benchmarking nRC dataset

To effectively evaluate the performance of the NCC, Non-Coding RNA Classification model proposed in this research, a comparative analysis with pre-existing non-coding RNA classification tools is essential. For a fair and accurate comparison, it's crucial to use an identical dataset for both training and testing across all tools. This ensures that the results are directly comparable and not influenced by variations in data. In this section, the NCC model, along with other established tools such as RNAcon, GraPPLE, nRC, ncRFP, ncDLRES, and ncDENSE, will be compared using the same dataset. It's important to note that the NCC model has not been specifically tailored or optimized for this dataset (instead it was trained and adjusted using the NCC dataset produced in this research), providing an unbiased basis for comparison. This approach will allow for a comprehensive evaluation of the NCC model's capabilities in the context of existing methodologies in non-coding RNA classification.

Model/Method	Accuracy	Sensitivity	Precision	F-score	MCC
RNAcon	0.3737	0.3787	0.4500	0.3605	0.3341
GraPPLE	0.6487	0.6684	0.7325	0.7050	0.6857
nRC	0.6960	0.6889	0.6878	0.6878	0.6627
ncRFP	0.7972	0.7878	0.7904	0.7883	0.7714
ncDLRES	0.8430	0.8344	0.8419	0.8407	0.8335
ncDENSE	0.8687	0.8677	0.8703	0.8667	0.8574
NCC 4d	0.9269	0.9269	0.9286	0.9268	0.9210
NCC 8d	0.9292	0.9292	0.9311	0.9293	0.9234

Table 9: Models comparison.

The table 9 showcases the performance of each tool across five key metrics. 4d and 8d refer to 4 digits and 8 digits encoding per RNA bases. It is evident from the table that the NCC (Non-Coding RNA Classification) tool outperforms its counterparts in all these metrics. Notably, the NCC tool demonstrates a slight improvement in accuracy, exceeding the next best model, ncDENSE, by approximately 0.06. This highlights the enhanced capability of the NCC tool in accurately classifying non-coding RNA sequences. Additionally, the accompanying figure 28 presents the confusion matrix for the NCC model, which has been both trained and tested using the nRC dataset.

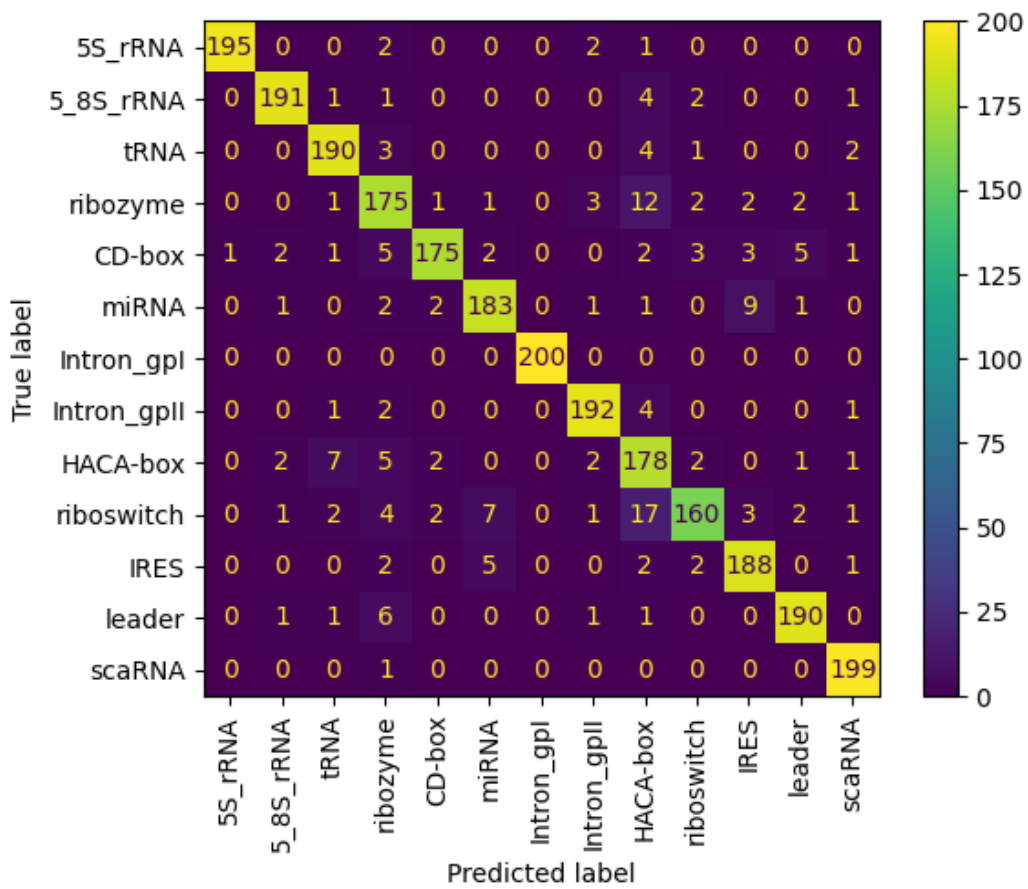


Figure 28: Confusion Matrix of NCC model train and tested with nRC dataset

8 Conclusion

This thesis is focused on creating and evaluating a tool, for classifying coding RNA (ncRNA) sequences. Aiming to develop an effective method that takes into account the intricate nature of ncRNAs, thereby contributing to the broader understanding of RNA biology and its implications in various biological processes and diseases.

Key features of this research are:

1. New dataset was developed (NCC dataset) ten times larger than the nRC dataset used in previous researches, providing a rich basis for training and evaluating the proposed classification model.
2. An in-depth analysis of two comprehensive ncRNA datasets, NCC and nRC, highlighting the diversity and complexity of ncRNA sequences.
3. The design and implementation of a unique neural network architecture, the NCC model, tailored specifically for the classification of ncRNAs. This model stands out by directly encoding RNA sequences and bypassing the potential inaccuracies of RNA secondary structure prediction tools.
4. The NCC model demonstrated superior performance, marginally outperforming existing top-tier classifiers in accuracy when assessed with identical datasets. This underscores the model's efficacy and the advancements in its architecture and training process. The model achieved exceptional accuracy rates, reaching up to 98% with the newly developed dataset, highlighting the significant progress made in ncRNA classification.

The model introduced in this study has demonstrated a notable performance, slightly surpassing the previously established top-performing classifiers when evaluated using identical training and testing datasets. Moreover, This achievement underscores the model's effectiveness and the advancements made in its design and implementation. Additionally, the model exhibits exceptional accuracy, achieving rates as high as 98% when utilizing the extended dataset that was developed during this research.

At present, RNA databases are somewhat underdeveloped and exhibit a significant imbalance among RNA classes. Certain non-coding RNA (ncRNA) classes are represented by a minimal number of sequences compared to others, underscoring a disparity in the availability of data. This scenario emphasizes that the domain of ncRNA classification remains a vibrant and promising field of research. There is a strong anticipation for the emergence of more sophisticated and precise methodologies for ncRNA classification. The ongoing interest and active research in this area suggest that we can expect the development of novel approaches that will significantly enhance the accuracy and effectiveness of ncRNA identification and classification.

9 Future Work

While the proposed classification model didn't incorporate secondary structure and graph properties as part of its input, it's important to highlight their potential significance. These elements, though not directly used in the current model, could offer valuable insights and enhance the model's predictive accuracy. Exploring their integration in future iterations of the model might uncover deeper relationships and patterns within the data, potentially leading to more robust and accurate classifications. Their relevance in the broader context of the field suggests they are worthy of further investigation and consideration.

A possible future approach is to develop a second parallel model for prediction non-coding RNA classes by using the graph features of the input RNA sequence. Along with other features extracted like the length of the sequence, k-mers substrings presence (as presented in [43]) and other RNA sequence extracted features. A possible architecture of a future model for ncRNA classification is shown in Figure 29.

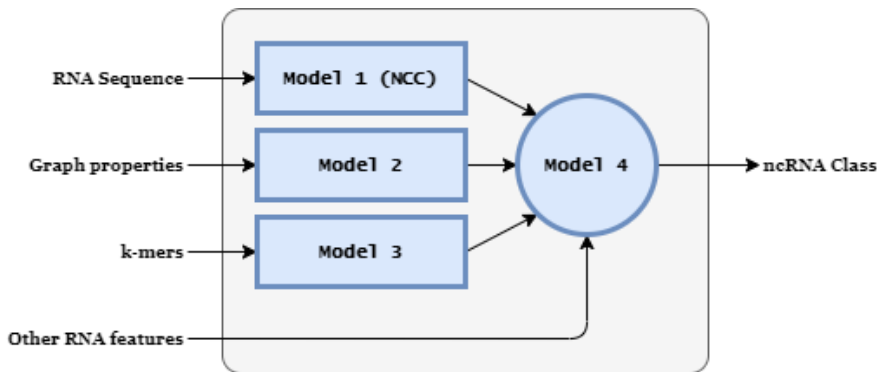


Figure 29: Proposed model for future work

A pivotal element of this research has been the development of the dataset, which holds significant weight in the overall effectiveness of our work. The quality of the dataset used for training is intrinsically linked to the accuracy of the model, directly influencing its capacity to produce precise predictions for new, unseen data. Recognizing this critical relationship, a key recommendation for future research is the creation of a comprehensive collection of generic non-coding RNA (ncRNA) datasets. These datasets would serve a dual purpose: firstly, as a resource for training and testing ncRNA classification tools, and secondly, as a benchmark for comparing the performance of various classifiers.

The development of such a collection would entail gathering a diverse range of ncRNA sequences, ensuring a wide representation of different ncRNA types. This diversity is essential to train models that are robust and capable of handling the complexity and variability inherent in ncRNA sequences. Furthermore, by standardizing the datasets used across different tools, we can facilitate a more objective and consistent comparison of their performance. Investment in these datasets would significantly enhance the field of ncRNA research, providing researchers with valuable resources to develop more advanced and accurate classification tools.

An alternative model that holds considerable potential features a collection of binary classifiers operating in parallel, with each classifier dedicated to identifying a specific class of non-coding RNA. The outputs of these individual classifiers are then aggregated by a final, overarching model, as presented in Figure 30. This master model analyzes the collective inputs from the binary classifiers to determine the specific class of the RNA sequence in question, effectively leveraging the strengths of each binary classifier to achieve a more nuanced and accurate classification outcome. This architecture allows for a focused approach where each binary classifier specializes in distinguishing its assigned RNA class, enhancing the overall precision and reliability of the RNA classification process.

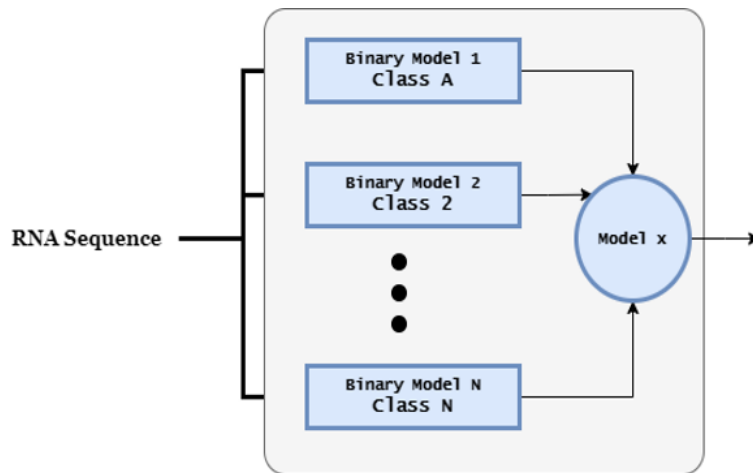


Figure 30: Proposed model for future work, consisting of multiple binary classifiers

Bibliography

- [1] Qi Zhao, Zheng Zhao, Xiaoya Fan, Zhengwei Yuan, Qian Mao, and Yudong Yao. Review of machine learning methods for rna secondary structure prediction. *PLoS computational biology*, 17:e1009291, 08 2021. doi: 10.1371/journal.pcbi.1009291.
- [2] Christos Andrikos, Evangelos Makris, Angelos Kolaitis, Georgios Rassias, Christos Pavlatos, and Panayiotis Tsanakas. Knotify: An efficient parallel platform for rna pseudoknot prediction using syntactic pattern recognition. *Methods and Protocols*, 5(1), 2022. ISSN 2409-9279. doi: 10.3390/mps5010014. URL <https://www.mdpi.com/2409-9279/5/1/14>.
- [3] Linyu Wang, Shaoge Zheng, Hao Zhang, Zhiyang Qiu, Xiaodan Zhong, Haiming Liu, and Yuanning Liu. ncrfp: A novel end-to-end method for non-coding rnas family prediction based on deep learning. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 18(2):784–789, 2021. doi: 10.1109/TCBB.2020.2982873.
- [4] Antonino Fiannaca, Massimo La Rosa, Laura La Paglia, Riccardo Rizzo, and Alfonso Urso. nrc: non-coding rna classifier based on structural features. *Bio-Data Mining*, 10, 2017. doi: <https://doi.org/10.1186/s13040-017-0148-2>. URL <https://doi.org/10.1186/s13040-017-0148-2>.
- [5] Alexander Hüttenhofer and Jörg Vogel. Experimental approaches to identify non-coding RNAs. *Nucleic Acids Research*, 34(2):635–646, 01 2006. ISSN 0305-1048. doi: 10.1093/nar/gkj469. URL <https://doi.org/10.1093/nar/gkj469>.
- [6] Feng Da-Fei and Doolittle Russell F. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *Journal of Molecular Evolution*, 25(4):351–360, 1987. doi: 10.1007/BF02603120. URL <https://doi.org/10.1007/BF02603120>.
- [7] John S. Mattick and Igor V. Makunin. Non-coding RNA. *Human Molecular Genetics*, 15(suppl_1):R17–R29, 04 2006. ISSN 0964-6906. doi: 10.1093/hmg/ddl046. URL <https://doi.org/10.1093/hmg/ddl046>.
- [8] Sam Griffiths-Jones, Alex Bateman, Mhairi Marshall, Ajay Khanna, and Sean R. Eddy. Rfam: an RNA family database. *Nucleic Acids Research*, 31(1):439–441, 01 2003. ISSN 0305-1048. doi: 10.1093/nar/gkg006. URL <https://doi.org/10.1093/nar/gkg006>.
- [9] Ioanna Kalvari, Joanna Argasinska, Natalia Quinones-Olvera, Eric P Nawrocki, Elena Rivas, Sean R Eddy, Alex Bateman, Robert D Finn, and Anton I Petrov. Rfam 13.0: shifting to a genome-centric resource for non-coding RNA families. *Nucleic Acids Research*, 46(D1):D335–D342, 11 2017. ISSN 0305-1048. doi: 10.1093/nar/gkx1038. URL <https://doi.org/10.1093/nar/gkx1038>.

- [10] RNAcentral Consortium. RNAcentral 2021: secondary structure integration, improved sequence search and new member databases. *Nucleic Acids Research*, 49(D1):D212–D220, 10 2020. ISSN 0305-1048. doi: 10.1093/nar/gkaa921. URL <https://doi.org/10.1093/nar/gkaa921>.
- [11] Paul G Higgs. Rna secondary structure: physical and computational aspects. *Quarterly reviews of biophysics*, 33(3):199–253, 2000.
- [12] Bharat Panwar, Amit Arora, and Gajendra PS Raghava. Prediction and classification of ncnas using structural information. *BMC Genomics*, 15, 2014. ISSN 1471-2164. doi: <https://doi.org/10.1186/1471-2164-15-127>. URL <https://doi.org/10.1186/1471-2164-15-127>.
- [13] 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(13):i85–i93, 06 2011. ISSN 1367-4803. doi: 10.1093/bioinformatics/btr215. URL <https://doi.org/10.1093/bioinformatics/btr215>.
- [14] Siddharth Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *International Journal of Engineering Applied Sciences and Technology*, 04:310–316, 05 2020. doi: 10.33564/IJEAST.2020.v04i12.054.
- [15] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL <https://doi.org/10.1162/neco.1997.9.8.1735>.
- [16] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.
- [17] Mike Schuster and Kuldeep Paliwal. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, 45:2673 – 2681, 12 1997. doi: 10.1109/78.650093.
- [18] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL <https://www.tensorflow.org/>. Software available from tensorflow.org.
- [19] François Chollet et al. Keras. <https://keras.io>, 2015.

- [20] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.
- [21] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL <https://doi.org/10.1038/s41586-020-2649-2>.
- [22] Jeff Reback, Wes McKinney, jbrockmendel, Joris Van den Bossche, Tom Augspurger, Phillip Cloud, gyoung, Sinhrks, Adam Klein, Matthew Roeschke, Jeff Tratner, Chang She, William Ayd, Simon Hawkins, Terji Petersen, Jeremy Schendel, Andy Hayden, Marc Garcia, Vytautas Jancauskas, MomIsBestFriend, Pietro Battiston, Skipper Seabold, chris b1, h vetinari, Stephan Hoyer, Wouter Overmeire, alimcmaster1, Mortada Mehyar, Christopher Whelan, and Thomas Kluyver. pandas-dev/pandas: Pandas 1.0.0. *Zenodo*, January 2020. doi: 10.5281/zenodo.3630805. URL <https://doi.org/10.5281/zenodo.3630805>.
- [23] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.
- [24] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.
- [25] Hosna Jabbari, Ian Wark, Carlo Montemagno, and Sebastian Will. Knotty: efficient and accurate prediction of complex RNA pseudoknot structures. *Bioinformatics*, 34(22):3849–3856, 06 2018. ISSN 1367-4803. doi: 10.1093/bioinformatics/bty420. URL <https://doi.org/10.1093/bioinformatics/bty420>.
- [26] Evangelos Makris, Angelos Kolaitis, Christos Andrikos, Vrettos Moulos, Panayiotis Tsanakas, and Christos Pavlatos. Knotify+: Toward the prediction of rna h-type pseudoknots, including bulges and internal loops. *Biomolecules*, 13(2), 2023. ISSN 2218-273X. doi: 10.3390/biom13020308. URL <https://www.mdpi.com/2218-273X/13/2/308>.
- [27] Evangelos Makris, Angelos Kolaitis, Christos Andrikos, Vrettos Moulos, Panayiotis Tsanakas, and Christos Pavlatos. An intelligent grammar-based platform for rna h-type pseudoknot prediction. In Ilias Maglogiannis, Lazaros Iliadis, John Macintyre, and Paulo Cortez, editors, *Artificial Intelligence Applications and Innovations. AIAI 2022 IFIP WG 12.5 International Workshops*, pages

- 174–186, Cham, 2022. Springer International Publishing. ISBN 978-3-031-08341-9.
- [28] Christos Koroulis, Evangelos Makris, Angelos Kolaitis, Panayiotis Tsanakas, and Christos Pavlatos. Syntactic pattern recognition for the prediction of l-type pseudoknots in rna. *Applied Sciences*, 13(8), 2023. ISSN 2076-3417. doi: 10.3390/app13085168. URL <https://www.mdpi.com/2076-3417/13/8/5168>.
- [29] Christian Borgelt, Thorsten Meinl, and Michael Berthold. Moss: A program for molecular substructure mining. In *Proceedings of the 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations*, OSDM '05, page 6–15, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595932100. doi: 10.1145/1133905.1133908. URL <https://doi.org/10.1145/1133905.1133908>.
- [30] Liam Childs, Zoran Nikoloski, Patrick May, and Dirk Walther. Identification and classification of ncRNA molecules using graph properties. *Nucleic Acids Research*, 37(9):e66–e66, 04 2009. ISSN 0305-1048. doi: 10.1093/nar/gkp206. URL <https://doi.org/10.1093/nar/gkp206>.
- [31] Linyu Wang, Xiaodan Zhong, Shuo Wang, and Yuanning Liu. ncdres: a novel method for non-coding rnas family prediction based on dynamic lstm and resnet. *BMC Bioinformatics*, 22, 09 2021. doi: 10.1186/s12859-021-04365-4. URL <https://doi.org/10.1186/s12859-021-04365-4>.
- [32] Kai Chen, Xiaodong Zhu, Lei Hao, Jiahao Wang, Zhen Liu, and Yuanning Liu. ncdense: a novel computational method based on a deep learning framework for non-coding rnas family prediction. *BMC Genomics*, 12 2022. doi: 10.21203/rs.3.rs-2374139/v1.
- [33] Jian Zhao, Yan Li, Cong Wang, Haotian Zhang, Hao Zhang, Bin Jiang, Xuejiang Guo, and Xiaofeng Song. Iresbase: A comprehensive database of experimentally validated internal ribosome entry sites. *Genomics, Proteomics & Bioinformatics*, 18(2):129–139, 2020. ISSN 1672-0229. doi: <https://doi.org/10.1016/j.gpb.2020.03.001>. URL <https://www.sciencedirect.com/science/article/pii/S1672022920300577>. Special Issue:Bioinformatics Commons—2020.
- [34] Ana Kozomara, Maria Birgaoanu, and Sam Griffiths-Jones. miRBase: from microRNA sequences to function. *Nucleic Acids Research*, 47(D1):D155–D162, 11 2018. ISSN 0305-1048. doi: 10.1093/nar/gky1141. URL <https://doi.org/10.1093/nar/gky1141>.
- [35] Changning Liu, Baoyan Bai, Geir Skogerbø, Lun Cai, Wei Deng, Yong Zhang, Dongbo Bu, Yi Zhao, and Runsheng Chen. NONCODE: an integrated knowledge database of non-coding RNAs. *Nucleic Acids Research*, 33(suppl_1): D112–D115, 01 2005. ISSN 0305-1048. doi: 10.1093/nar/gki041. URL <https://doi.org/10.1093/nar/gki041>.

- [36] Fergal J Martin, M Ridwan Amode, Alisha Aneja, Olanrewaju Austine-Orimoloye, Andrey G Azov, If Barnes, Arne Becker, Ruth Bennett, Andrew Berry, Jyothish Bhai, Simarpreet Kaur Bhurji, Alexandra Bignell, Sanjay Boddu, Paulo R Branco Lins, Lucy Brooks, Shashank Budhanuru Ramaraju, Mehrnaz Charkhchi, Alexander Cockburn, Luca Da Rin Fiorretto, Claire Davidson, Kamalkumar Dodiya, Sarah Donaldson, Bilal El Houdaigui, Tamara El Naboulsi, Reham Fatima, Carlos Garcia Giron, Thiago Genez, Gurpreet S Ghattaoraya, Jose Gonzalez Martinez, Cristi Guijarro, Matthew Hardy, Zoe Hollis, Thibaut Hourlier, Toby Hunt, Mike Kay, Vinay Kaykala, Tuan Le, Diana Lemos, Diego Marques-Coelho, José Carlos Marugán, Gabriela Alejandra Merino, Louisse Paola Mirabueno, Aleena Mushtaq, Syed Nakib Hosain, Denye N Ogeh, Manoj Pandian Sakthivel, Anne Parker, Malcolm Perry, Ivana Piližota, Irina Prosovetskaia, José G Pérez-Silva, Ahamed Imran Abdul Salam, Nuno Saraiva-Agostinho, Helen Schuilenburg, Dan Sheppard, Swati Sinha, Botond Sipos, William Stark, Emily Steed, Ranjit Sukumaran, Dulika Sumathipala, Marie-Marthe Suner, Likhitha Surapaneni, Kyösti Sutinen, Michal Szpak, Francesca Floriana Tricomi, David Urbina-Gómez, Andres Veidenberg, Thomas A Walsh, Brandon Walts, Elizabeth Wass, Natalie Willhoft, Jamie Allen, Jorge Alvarez-Jarreta, Marc Chakiachvili, Bethany Flint, Stefano Giorgetti, Leanne Haggerty, Garth R Ilsley, Jane E Loveland, Benjamin Moore, Jonathan M Mudge, John Tate, David Thybert, Stephen J Trevanion, Andrea Winterbottom, Adam Frankish, Sarah E Hunt, Magali Ruffier, Fiona Cunningham, Sarah Dyer, Robert D Finn, Kevin L Howe, Peter W Harrison, Andrew D Yates, and Paul Flicek. Ensembl 2023. *Nucleic Acids Research*, 51 (D1):D933–D941, 11 2022. ISSN 0305-1048. doi: 10.1093/nar/gkac958. URL <https://doi.org/10.1093/nar/gkac958>.
- [37] Gil Stelzer, Naomi Rosen, Inbar Plaschkes, Shahar Zimmerman, Michal Twik, Simon Fishilevich, Tsippi Iny Stein, Ron Nudel, Iris Lieder, Yaron Mazor, Sergey Kaplan, Dvir Dahary, David Warshawsky, Yaron Guan-Golan, Asher Kohn, Noa Rappaport, Marilyn Safran, and Doron Lancet. The genecards suite: From gene data mining to disease genome sequence analyses. *Current Protocols in Bioinformatics*, 54(1):1.30.1–1.30.33, 2016. doi: <https://doi.org/10.1002/cpbi.5>. URL <https://currentprotocols.onlinelibrary.wiley.com/doi/abs/10.1002/cpbi.5>.
- [38] Pieter-Jan Volders, Jasper Anckaert, Kenneth Verheggen, Justine Nuytens, Lennart Martens, Pieter Mestdagh, and Jo Vandesompele. LNCipedia 5: towards a reference set of human long non-coding RNAs. *Nucleic Acids Research*, 47(D1):D135–D139, 10 2018. ISSN 0305-1048. doi: 10.1093/nar/gky1031. URL <https://doi.org/10.1093/nar/gky1031>.
- [39] Catherine Brooksbank, Evelyn Camon, Midori A. Harris, Michele Magrane, Maria Jesus Martin, Nicola Mulder, Claire O’Donovan, Helen Parkinson, Mary Ann Tuli, Rolf Apweiler, Ewan Birney, Alvis Brazma, Kim Henrick, Rodrigo Lopez, Guenter Stoesser, Peter Stoehr, and Graham Cameron. The European Bioinformatics Institute’s data resources. *Nucleic Acids Research*,

31(1):43–50, 01 2003. ISSN 0305-1048. doi: 10.1093/nar/gkg066. URL <https://doi.org/10.1093/nar/gkg066>.

- [40] Ioanna Kalvari, Eric P. Nawrocki, Joanna Argasinska, Natalia Quinones-Olvera, Robert D. Finn, Alex Bateman, and Anton I. Petrov. Non-coding rna analysis using the rfam database. *Current Protocols in Bioinformatics*, 62(1):e51, 2018. doi: <https://doi.org/10.1002/cpbi.51>. URL <https://currentprotocols.onlinelibrary.wiley.com/doi/abs/10.1002/cpbi.51>.
- [41] Limin Fu, Beifang Niu, Zhengwei Zhu, Sitao Wu, and Weizhong Li. CD-HIT: accelerated for clustering the next-generation sequencing data. *Bioinformatics*, 28(23):3150–3152, 10 2012. ISSN 1367-4803. doi: 10.1093/bioinformatics/bts565. URL <https://doi.org/10.1093/bioinformatics/bts565>.
- [42] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.
- [43] Chandran Nithin, Sunandan Mukherjee, Jolly Basak, and Ranjit Prasad Bahadur. Ncodr: A multi-class support vector machine classification to distinguish non-coding rnas in viridiplantae. *Quantitative Plant Biology*, 3:e23, 2022. doi: 10.1017/qpb.2022.18.